# Oracle® Database

Reference

12*c* Release 1 (12.1)

**E41527-20**

June 2015

ORACLE®

Oracle Database Reference, 12c Release 1 (12.1)

E41527-20

Copyright © 2002, 2015, Oracle and/or its affiliates. All rights reserved.

Primary Author: Bert Rich

Contributors: Nipun Agarwal, Rick Anderson, Mark Bauer, Neerja Bhatt, Tudor Bosman, George Candea, Wilson Chan, Sumanta Chatterjee, Eugene Chong, Connie Dialeris Green, Harvey Eneman, Bruce Ernst, Ira Greenberg, Steve Harris, Thuvan Hoang, Lilian Hobbs, Namit Jain, Hakkan Jakobsson, Bob Jenkins, Mark Johnson, Jonathan Klein, Sushil Kumar, Tirthankar Lahiri, Bill Lee, Yunrui Li, Juan Loaiza, Rich Long, Diana Lorentz, Catherine Luu, Neil MacNaughton, Ari Mozes, Gopal Mulagund, Subramanian Muralidhar, Ravi Murthy, Karuna Muthiah, Arvind Nithrakashyap, Ronald Obermarck, Kant Patel, Christopher Racicot, Mark Ramacher, Shankar Raman, Suvendu Ray, Ann Rhee, Beth Roeser, Kathy Rich, Vivian Schupmann, Debbie Steiner, Malai Stalin, Seema Sundara, Juan Tellez, Mark Townsend, Alex Tsukerman, Kothanda Umamageswaran, Randy Urbano, Steve Vivian, Eric Voss, Tak Wang, Steve Wertheimer, Andy Witkowski, Daniel Wong, Graham Wood, Brian Wright, Aravind Yalamanchi, Qin Yu, Mohamed Ziauddin

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

# Contents


Preface ..... lxvii

Audience ..... lxvii
Documentation Accessibility ..... lxviii
Related Documents ..... lxviii
Conventions ..... lxviii

Changes in This Release for Oracle Database Reference ..... lxix

Changes in Oracle Database 12c Release 1 (12.1.0.2) ..... lxix
New Features ..... lxix
Deprecated Features ..... lxxi
Changes in Oracle Database 12c Release 1 (12.1.0.1) ..... lxxi
New Features ..... lxxi
Deprecated Features ..... lxxxi

**Part I** Initialization Parameters

**1** Initialization Parameters

Uses of Initialization Parameters ..... 1-1
Types of Initialization Parameters ..... 1-2
Basic Initialization Parameters ..... 1-3
Parameter Files ..... 1-3
Server Parameter Files ..... 1-3
Initialization Parameter Files ..... 1-4
Changing Parameter Values in a Parameter File ..... 1-7
Parameters by Functional Category ..... 1-7
Modifiable Parameters ..... 1-14
Displaying Current Parameter Values ..... 1-21
Parameters You Should Not Specify in the Parameter File ..... 1-21
When Parameters Are Set Incorrectly ..... 1-21
Reading the Parameter Descriptions ..... 1-21
Initialization Parameter Descriptions ..... 1-22

ACTIVE_INSTANCE_COUNT ..... 1-23
AQ_TM_PROCESSES ..... 1-23
ARCHIVE_LAG_TARGET ..... 1-24
ASM_DISKGROUPS ..... 1-25
ASM_DISKSTRING ..... 1-25
ASM_POWER_LIMIT ..... 1-26
ASM_PREFERRED_READ_FAILURE_GROUPS ..... 1-27
AUDIT_FILE_DEST ..... 1-27
AUDIT_SYS_OPERATIONS ..... 1-28
AUDIT_SYSLOG_LEVEL ..... 1-29
AUDIT_TRAIL ..... 1-30
AWR_SNAPSHOT_TIME_OFFSET ..... 1-33
BACKGROUND_CORE_DUMP ..... 1-33
BACKGROUND_DUMP_DEST ..... 1-34
BACKUP_TAPE_IO_SLAVES ..... 1-35
BITMAP_MERGE_AREA_SIZE ..... 1-35
BLANK_TRIMMING ..... 1-36
CIRCUITS ..... 1-37
CLIENT_RESULT_CACHE_LAG ..... 1-37
CLIENT_RESULT_CACHE_SIZE ..... 1-38
CLONEDB ..... 1-38
CLUSTER_DATABASE ..... 1-39
CLUSTER_DATABASE_INSTANCES ..... 1-39
CLUSTER_INTERCONNECTS ..... 1-40
COMMIT_LOGGING ..... 1-41
COMMIT_POINT_STRENGTH ..... 1-41
COMMIT_WAIT ..... 1-42
COMMIT_WRITE ..... 1-42
COMMON_USER_PREFIX ..... 1-43
COMPATIBLE ..... 1-44
CONNECTION_BROKERS ..... 1-45
CONTROL_FILE_RECORD_KEEP_TIME ..... 1-46
CONTROL_FILES ..... 1-47
CONTROL_MANAGEMENT_PACK_ACCESS ..... 1-48
CORE_DUMP_DEST ..... 1-48
CPU_COUNT ..... 1-49
CREATE_BITMAP_AREA_SIZE ..... 1-50
CREATE_STORED_OUTLINES ..... 1-50
CURSOR_BIND_CAPTURE_DESTINATION ..... 1-51
CURSOR_SHARING ..... 1-52
CURSOR_SPACE_FOR_TIME ..... 1-53
DB_nK_CACHE_SIZE ..... 1-54
DB_BIG_TABLE_CACHE_PERCENT_TARGET ..... 1-54

DB_BLOCK_BUFFERS ... 1-56
DB_BLOCK_CHECKING ... 1-57
DB_BLOCK_CHECKSUM ... 1-58
DB_BLOCK_SIZE ... 1-59
DB_CACHE_ADVICE ... 1-59
DB_CACHE_SIZE ... 1-60
DB_CREATE_FILE_DEST ... 1-61
DB_CREATE_ONLINE_LOG_DEST_n ... 1-62
DB_DOMAIN ... 1-62
DB_FILE_MULTIBLOCK_READ_COUNT ... 1-63
DB_FILE_NAME_CONVERT ... 1-64
DB_FILES ... 1-65
DB_FLASH_CACHE_FILE ... 1-66
DB_FLASH_CACHE_SIZE ... 1-66
DB_FLASHBACK_RETENTION_TARGET ... 1-67
DB_INDEX_COMPRESSION_INHERITANCE ... 1-68
DB_KEEP_CACHE_SIZE ... 1-68
DB_LOST_WRITE_PROTECT ... 1-69
DB_NAME ... 1-70
DB_RECOVERY_FILE_DEST ... 1-70
DB_RECOVERY_FILE_DEST_SIZE ... 1-71
DB_RECYCLE_CACHE_SIZE ... 1-72
DB_SECUREFILE ... 1-72
DB_ULTRA_SAFE ... 1-74
DB_UNIQUE_NAME ... 1-74
DB_UNRECOVERABLE_SCN_TRACKING ... 1-75
DB_WRITER_PROCESSES ... 1-76
DBFIPS_140 ... 1-76
DBWR_IO_SLAVES ... 1-77
DDL_LOCK_TIMEOUT ... 1-78
DEFERRED_SEGMENT_CREATION ... 1-78
DG_BROKER_CONFIG_FILEn ... 1-79
DG_BROKER_START ... 1-80
DIAGNOSTIC_DEST ... 1-80
DISK_ASYNCH_IO ... 1-81
DISPATCHERS ... 1-82
DISTRIBUTED_LOCK_TIMEOUT ... 1-84
DML_LOCKS ... 1-84
DNFS_BATCH_SIZE ... 1-85
DST_UPGRADE_INSERT_CONV ... 1-86
ENABLE_DDL_LOGGING ... 1-87
ENABLE_GOLDENGATE_REPLICATION ... 1-88
ENABLE_PLUGGABLE_DATABASE ... 1-89

EVENT ... 1-90
EXAFUSION_ENABLED ... 1-90
FAL_CLIENT ... 1-91
FAL_SERVER ... 1-91
FAST_START_MTTR_TARGET ... 1-92
FAST_START_PARALLEL_ROLLBACK ... 1-92
FILE_MAPPING ... 1-93
FILEIO_NETWORK_ADAPTERS ... 1-94
FILESYSTEMIO_OPTIONS ... 1-94
FIXED_DATE ... 1-95
GCS_SERVER_PROCESSES ... 1-95
GLOBAL_CONTEXT_POOL_SIZE ... 1-96
GLOBAL_NAMES ... 1-96
GLOBAL_TXN_PROCESSES ... 1-97
HASH_AREA_SIZE ... 1-97
HEAT_MAP ... 1-98
HI_SHARED_MEMORY_ADDRESS ... 1-99
HS_AUTOREGISTER ... 1-99
IFILE ... 1-99
INMEMORY_CLAUSE_DEFAULT ... 1-100
INMEMORY_FORCE ... 1-105
INMEMORY_MAX_POPULATE_SERVERS ... 1-106
INMEMORY_QUERY ... 1-107
INMEMORY_SIZE ... 1-108
INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT ... 1-109
INSTANCE_GROUPS ... 1-110
INSTANCE_NAME ... 1-110
INSTANCE_NUMBER ... 1-111
INSTANCE_TYPE ... 1-112
JAVA_JIT_ENABLED ... 1-112
JAVA_MAX_SESSIONSPACE_SIZE ... 1-113
JAVA_POOL_SIZE ... 1-113
JAVA_SOFT_SESSIONSPACE_LIMIT ... 1-114
JOB_QUEUE_PROCESSES ... 1-115
LARGE_POOL_SIZE ... 1-115
LDAP_DIRECTORY_ACCESS ... 1-117
LDAP_DIRECTORY_SYSAUTH ... 1-118
LICENSE_MAX_SESSIONS ... 1-118
LICENSE_MAX_USERS ... 1-119
LICENSE_SESSIONS_WARNING ... 1-120
LISTENER_NETWORKS ... 1-121
LOCAL_LISTENER ... 1-121
LOCK_NAME_SPACE ... 1-122

LOCK_SGA ... 1-122
LOG_ARCHIVE_CONFIG ... 1-123
LOG_ARCHIVE_DEST ... 1-124
LOG_ARCHIVE_DEST_n ... 1-125
LOG_ARCHIVE_DEST_STATE_n ... 1-129
LOG_ARCHIVE_DUPLEX_DEST ... 1-130
LOG_ARCHIVE_FORMAT ... 1-131
LOG_ARCHIVE_MAX_PROCESSES ... 1-133
LOG_ARCHIVE_MIN_SUCCEED_DEST ... 1-133
LOG_ARCHIVE_TRACE ... 1-134
LOG_BUFFER ... 1-137
LOG_CHECKPOINT_INTERVAL ... 1-138
LOG_CHECKPOINT_TIMEOUT ... 1-139
LOG_CHECKPOINTS_TO_ALERT ... 1-139
LOG_FILE_NAME_CONVERT ... 1-140
MAX_DISPATCHERS ... 1-141
MAX_DUMP_FILE_SIZE ... 1-141
MAX_ENABLED_ROLES ... 1-142
MAX_SHARED_SERVERS ... 1-143
MAX_STRING_SIZE ... 1-144
MEMORY_MAX_TARGET ... 1-147
MEMORY_TARGET ... 1-147
NLS_CALENDAR ... 1-148
NLS_COMP ... 1-149
NLS_CURRENCY ... 1-150
NLS_DATE_FORMAT ... 1-151
NLS_DATE_LANGUAGE ... 1-152
NLS_DUAL_CURRENCY ... 1-153
NLS_ISO_CURRENCY ... 1-153
NLS_LANGUAGE ... 1-154
NLS_LENGTH_SEMANTICS ... 1-155
NLS_NCHAR_CONV_EXCP ... 1-156
NLS_NUMERIC_CHARACTERS ... 1-157
NLS_SORT ... 1-157
NLS_TERRITORY ... 1-159
NLS_TIMESTAMP_FORMAT ... 1-159
NLS_TIMESTAMP_TZ_FORMAT ... 1-160
NONCDB_COMPATIBLE ... 1-161
O7_DICTIONARY_ACCESSIBILITY ... 1-162
OBJECT_CACHE_MAX_SIZE_PERCENT ... 1-163
OBJECT_CACHE_OPTIMAL_SIZE ... 1-163
OLAP_PAGE_POOL_SIZE ... 1-164
OPEN_CURSORS ... 1-164

OPEN_LINKS ... 1-165
OPEN_LINKS_PER_INSTANCE ... 1-166
OPTIMIZER_ADAPTIVE_FEATURES ... 1-166
OPTIMIZER_ADAPTIVE_REPORTING_ONLY ... 1-167
OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES ... 1-167
OPTIMIZER_DYNAMIC_SAMPLING ... 1-168
OPTIMIZER_FEATURES_ENABLE ... 1-169
OPTIMIZER_INDEX_CACHING ... 1-171
OPTIMIZER_INDEX_COST_ADJ ... 1-172
OPTIMIZER_INMEMORY_AWARE ... 1-172
OPTIMIZER_MODE ... 1-173
OPTIMIZER_SECURE_VIEW_MERGING ... 1-174
OPTIMIZER_USE_INVISIBLE_INDEXES ... 1-175
OPTIMIZER_USE_PENDING_STATISTICS ... 1-175
OPTIMIZER_USE_SQL_PLAN_BASELINES ... 1-175
OS_AUTHENT_PREFIX ... 1-176
OS_ROLES ... 1-177
PARALLEL_ADAPTIVE_MULTI_USER ... 1-177
PARALLEL_AUTOMATIC_TUNING ... 1-178
PARALLEL_DEGREE_LIMIT ... 1-179
PARALLEL_DEGREE_POLICY ... 1-180
PARALLEL_EXECUTION_MESSAGE_SIZE ... 1-181
PARALLEL_FORCE_LOCAL ... 1-182
PARALLEL_INSTANCE_GROUP ... 1-182
PARALLEL_IO_CAP_ENABLED ... 1-183
PARALLEL_MAX_SERVERS ... 1-184
PARALLEL_MIN_PERCENT ... 1-185
PARALLEL_MIN_SERVERS ... 1-186
PARALLEL_MIN_TIME_THRESHOLD ... 1-187
PARALLEL_SERVERS_TARGET ... 1-187
PARALLEL_THREADS_PER_CPU ... 1-188
PDB_FILE_NAME_CONVERT ... 1-189
PERMIT_92_WRAP_FORMAT ... 1-190
PGA_AGGREGATE_LIMIT ... 1-190
PGA_AGGREGATE_TARGET ... 1-191
PLSCOPE_SETTINGS ... 1-192
PLSQL_CCFLAGS ... 1-193
PLSQL_CODE_TYPE ... 1-194
PLSQL_DEBUG ... 1-194
PLSQL_OPTIMIZE_LEVEL ... 1-195
PLSQL_V2_COMPATIBILITY ... 1-196
PLSQL_WARNINGS ... 1-197
PRE_PAGE_SGA ... 1-198

PROCESSES ... 1-199
PROCESSOR_GROUP_NAME ... 1-199
QUERY_REWRITE_ENABLED ... 1-200
QUERY_REWRITE_INTEGRITY ... 1-201
RDBMS_SERVER_DN ... 1-202
READ_ONLY_OPEN_DELAYED ... 1-203
RECOVERY_PARALLELISM ... 1-203
RECYCLEBIN ... 1-204
REDO_TRANSPORT_USER ... 1-204
REMOTE_DEPENDENCIES_MODE ... 1-205
REMOTE_LISTENER ... 1-206
REMOTE_LOGIN_PASSWORDFILE ... 1-206
REMOTE_OS_AUTHENT ... 1-208
REMOTE_OS_ROLES ... 1-208
REPLICATION_DEPENDENCY_TRACKING ... 1-209
RESOURCE_LIMIT ... 1-209
RESOURCE_MANAGER_CPU_ALLOCATION ... 1-210
RESOURCE_MANAGER_PLAN ... 1-210
RESULT_CACHE_MAX_RESULT ... 1-212
RESULT_CACHE_MAX_SIZE ... 1-213
RESULT_CACHE_MODE ... 1-213
RESULT_CACHE_REMOTE_EXPIRATION ... 1-214
RESUMABLE_TIMEOUT ... 1-215
ROLLBACK_SEGMENTS ... 1-215
SEC_CASE_SENSITIVE_LOGON ... 1-216
SEC_MAX_FAILED_LOGIN_ATTEMPTS ... 1-217
SEC_PROTOCOL_ERROR_FURTHER_ACTION ... 1-217
SEC_PROTOCOL_ERROR_TRACE_ACTION ... 1-218
SEC_RETURN_SERVER_RELEASE_BANNER ... 1-218
SERIAL_REUSE ... 1-219
SERVICE_NAMES ... 1-220
SESSION_CACHED_CURSORS ... 1-221
SESSION_MAX_OPEN_FILES ... 1-222
SESSIONS ... 1-222
SGA_MAX_SIZE ... 1-223
SGA_TARGET ... 1-224
SHADOW_CORE_DUMP ... 1-225
SHARED_MEMORY_ADDRESS ... 1-226
SHARED_POOL_RESERVED_SIZE ... 1-226
SHARED_POOL_SIZE ... 1-227
SHARED_SERVER_SESSIONS ... 1-228
SHARED_SERVERS ... 1-228
SKIP_UNUSABLE_INDEXES ... 1-229

SMTP_OUT_SERVER ... 1-230
SORT_AREA_RETAINED_SIZE ... 1-231
SORT_AREA_SIZE ... 1-232
SPATIAL_VECTOR_ACCELERATION ... 1-233
SPFILE ... 1-233
SQL_TRACE ... 1-234
SQL92_SECURITY ... 1-235
SQLTUNE_CATEGORY ... 1-236
STANDBY_ARCHIVE_DEST ... 1-236
STANDBY_FILE_MANAGEMENT ... 1-237
STAR_TRANSFORMATION_ENABLED ... 1-237
STATISTICS_LEVEL ... 1-238
STREAMS_POOL_SIZE ... 1-240
TAPE_ASYNCH_IO ... 1-241
TEMP_UNDO_ENABLED ... 1-241
THREAD ... 1-242
THREADED_EXECUTION ... 1-242
TIMED_OS_STATISTICS ... 1-244
TIMED_STATISTICS ... 1-244
TRACE_ENABLED ... 1-245
TRACEFILE_IDENTIFIER ... 1-246
TRANSACTIONS ... 1-247
TRANSACTIONS_PER_ROLLBACK_SEGMENT ... 1-247
UNDO_MANAGEMENT ... 1-248
UNDO_RETENTION ... 1-248
UNDO_TABLESPACE ... 1-249
UNIFIED_AUDIT_SGA_QUEUE_SIZE ... 1-250
USE_DEDICATED_BROKER ... 1-250
USE_INDIRECT_DATA_BUFFERS ... 1-252
USE_LARGE_PAGES ... 1-252
USER_DUMP_DEST ... 1-253
UTL_FILE_DIR ... 1-254
WORKAREA_SIZE_POLICY ... 1-255
XML_DB_EVENTS ... 1-255

**Part II** Static Data Dictionary Views

**2** Static Data Dictionary Views: ALL_ALL_TABLES to ALL_OUTLINES

About Static Data Dictionary Views ... 2-1
CDB_* Views ... 2-2
Oracle Database Real Application Security Views ... 2-2
Oracle Label Security Views ... 2-3
Oracle Database Vault Views ... 2-3

Oracle Replication Views ... 2-3
Oracle Workspace Manager Views ... 2-5
Recovery Catalog Views ... 2-6
Static Data Dictionary View Descriptions ... 2-7
ALL_ALL_TABLES ... 2-7
ALL_APPLY ... 2-12
ALL_APPLY_CHANGE_HANDLERS ... 2-14
ALL_APPLY_CONFLICT_COLUMNS ... 2-15
ALL_APPLY_DML_HANDLERS ... 2-16
ALL_APPLY_ENQUEUE ... 2-17
ALL_APPLY_ERROR ... 2-17
ALL_APPLY_ERROR_MESSAGES ... 2-19
ALL_APPLY_EXECUTE ... 2-21
ALL_APPLY_INSTANTIATED_GLOBAL ... 2-21
ALL_APPLY_INSTANTIATED_OBJECTS ... 2-22
ALL_APPLY_INSTANTIATED_SCHEMAS ... 2-22
ALL_APPLY_KEY_COLUMNS ... 2-23
ALL_APPLY_PARAMETERS ... 2-23
ALL_APPLY_PROGRESS ... 2-24
ALL_APPLY_TABLE_COLUMNS ... 2-25
ALL_ARGUMENTS ... 2-26
ALL_ASSEMBLIES ... 2-28
ALL_ASSOCIATIONS ... 2-29
ALL_ATTRIBUTE_TRANSFORMATIONS ... 2-30
ALL_AUDIT_POLICIES ... 2-31
ALL_AUDIT_POLICY_COLUMNS ... 2-32
ALL_AW_PS ... 2-33
ALL_AWS ... 2-34
ALL_BASE_TABLE_MVIEWS ... 2-35
ALL_CAPTURE ... 2-35
ALL_CAPTURE_EXTRA_ATTRIBUTES ... 2-38
ALL_CAPTURE_PARAMETERS ... 2-39
ALL_CAPTURE_PREPARED_DATABASE ... 2-40
ALL_CAPTURE_PREPARED_SCHEMAS ... 2-40
ALL_CAPTURE_PREPARED_TABLES ... 2-41
ALL_CATALOG ... 2-42
ALL_CLUSTER_HASH_EXPRESSIONS ... 2-43
ALL_CLUSTERING_DIMENSIONS ... 2-43
ALL_CLUSTERING_JOINS ... 2-44
ALL_CLUSTERING_KEYS ... 2-45
ALL_CLUSTERING_TABLES ... 2-46
ALL_CLUSTERS ... 2-48
ALL_CODE_ROLE_PRIVS ... 2-50

ALL_COL_COMMENTS ... 2-50
ALL_COL_PENDING_STATS ... 2-51
ALL_COL_PRIVS ... 2-52
ALL_COL_PRIVS_MADE ... 2-53
ALL_COL_PRIVS_RECD ... 2-54
ALL_COLL_TYPES ... 2-54
ALL_CONS_COLUMNS ... 2-55
ALL_CONS_OBJ_COLUMNS ... 2-56
ALL_CONSTRAINTS ... 2-57
ALL_CONTEXT ... 2-59
ALL_CREDENTIALS ... 2-60
ALL_CUBE_ATTR_VISIBILITY ... 2-61
ALL_CUBE_ATTRIBUTES ... 2-62
ALL_CUBE_BUILD_PROCESSES ... 2-63
ALL_CUBE_CALCULATED_MEMBERS ... 2-64
ALL_CUBE_DIM_LEVELS ... 2-65
ALL_CUBE_DIM_MODELS ... 2-65
ALL_CUBE_DIM_VIEW_COLUMNS ... 2-66
ALL_CUBE_DIM_VIEWS ... 2-67
ALL_CUBE_DIMENSIONALITY ... 2-67
ALL_CUBE_DIMENSIONS ... 2-68
ALL_CUBE_HIER_LEVELS ... 2-69
ALL_CUBE_HIER_VIEW_COLUMNS ... 2-70
ALL_CUBE_HIER_VIEWS ... 2-71
ALL_CUBE_HIERARCHIES ... 2-72
ALL_CUBE_MEASURES ... 2-73
ALL_CUBE_NAMED_BUILD_SPECS ... 2-74
ALL_CUBE_SUB_PARTITION_LEVELS ... 2-75
ALL_CUBE_VIEW_COLUMNS ... 2-75
ALL_CUBE_VIEWS ... 2-76
ALL_CUBES ... 2-77
ALL_DB_LINKS ... 2-78
ALL_DEF_AUDIT_OPTS ... 2-78
ALL_DEPENDENCIES ... 2-80
ALL_DEQUEUE_QUEUES ... 2-81
ALL_DIM_ATTRIBUTES ... 2-81
ALL_DIM_CHILD_OF ... 2-82
ALL_DIM_HIERARCHIES ... 2-83
ALL_DIM_JOIN_KEY ... 2-83
ALL_DIM_LEVEL_KEY ... 2-84
ALL_DIM_LEVELS ... 2-84
ALL_DIMENSIONS ... 2-85
ALL_DIRECTORIES ... 2-86

ALL_EDITION_COMMENTS ..... 2-86
ALL_EDITIONING_VIEW_COLS ..... 2-87
ALL_EDITIONING_VIEW_COLS_AE ..... 2-87
ALL_EDITIONING_VIEWS ..... 2-88
ALL_EDITIONING_VIEWS_AE ..... 2-89
ALL_EDITIONS ..... 2-89
ALL_ENCRYPTED_COLUMNS ..... 2-90
ALL_ERROR_TRANSLATIONS ..... 2-90
ALL_ERRORS ..... 2-91
ALL_ERRORS_AE ..... 2-92
ALL_EVALUATION_CONTEXT_TABLES ..... 2-93
ALL_EVALUATION_CONTEXT_VARS ..... 2-94
ALL_EVALUATION_CONTEXTS ..... 2-94
ALL_EXTERNAL_LOCATIONS ..... 2-95
ALL_EXTERNAL_TABLES ..... 2-96
ALL_FILE_GROUP_EXPORT_INFO ..... 2-97
ALL_FILE_GROUP_FILES ..... 2-97
ALL_FILE_GROUP_TABLES ..... 2-98
ALL_FILE_GROUP_TABLESPACES ..... 2-99
ALL_FILE_GROUP_VERSIONS ..... 2-100
ALL_FILE_GROUPS ..... 2-100
ALL_GG_INBOUND_PROGRESS ..... 2-101
ALL_GOLDENGATE_INBOUND ..... 2-102
ALL_GOLDENGATE_PRIVILEGES ..... 2-103
ALL_GOLDENGATE_RULES ..... 2-104
ALL_HEAT_MAP_SEG_HISTOGRAM ..... 2-105
ALL_HEAT_MAP_SEGMENT ..... 2-106
ALL_HISTOGRAMS ..... 2-107
ALL_IDENTIFIERS ..... 2-107
ALL_IND_COLUMNS ..... 2-108
ALL_IND_EXPRESSIONS ..... 2-109
ALL_IND_PARTITIONS ..... 2-110
ALL_IND_PENDING_STATS ..... 2-113
ALL_IND_STATISTICS ..... 2-114
ALL_IND_SUBPARTITIONS ..... 2-116
ALL_INDEXES ..... 2-119
ALL_INDEXTYPE_ARRAYTYPES ..... 2-124
ALL_INDEXTYPE_COMMENTS ..... 2-125
ALL_INDEXTYPE_OPERATORS ..... 2-125
ALL_INDEXTYPES ..... 2-126
ALL_INTERNAL_TRIGGERS ..... 2-127
ALL_JAVA_ARGUMENTS ..... 2-127
ALL_JAVA_CLASSES ..... 2-128

ALL_JAVA_COMPILER_OPTIONS ... 2-129
ALL_JAVA_DERIVATIONS ... 2-130
ALL_JAVA_FIELDS ... 2-130
ALL_JAVA_IMPLEMENTS ... 2-132
ALL_JAVA_INNERS ... 2-132
ALL_JAVA_LAYOUTS ... 2-134
ALL_JAVA_METHODS ... 2-134
ALL_JAVA_NCOMPS ... 2-136
ALL_JAVA_RESOLVERS ... 2-137
ALL_JAVA_THROWS ... 2-137
ALL_JOBS ... 2-138
ALL_JOIN_IND_COLUMNS ... 2-138
ALL_JSON_COLUMNS ... 2-139
ALL_LIBRARIES ... 2-140
ALL_LOB_PARTITIONS ... 2-141
ALL_LOB_SUBPARTITIONS ... 2-143
ALL_LOB_TEMPLATES ... 2-146
ALL_LOBS ... 2-147
ALL_LOG_GROUP_COLUMNS ... 2-149
ALL_LOG_GROUPS ... 2-150
ALL_MEASURE_FOLDER_CONTENTS ... 2-151
ALL_MEASURE_FOLDER_SUBFOLDERS ... 2-151
ALL_MEASURE_FOLDERS ... 2-152
ALL_METADATA_PROPERTIES ... 2-153
ALL_METHOD_PARAMS ... 2-153
ALL_METHOD_RESULTS ... 2-154
ALL_MINING_MODEL_ATTRIBUTES ... 2-155
ALL_MINING_MODEL_SETTINGS ... 2-157
ALL_MINING_MODELS ... 2-157
ALL_MVIEW_AGGREGATES ... 2-159
ALL_MVIEW_ANALYSIS ... 2-160
ALL_MVIEW_COMMENTS ... 2-161
ALL_MVIEW_DETAIL_PARTITION ... 2-162
ALL_MVIEW_DETAIL_RELATIONS ... 2-163
ALL_MVIEW_DETAIL_SUBPARTITION ... 2-164
ALL_MVIEW_JOINS ... 2-165
ALL_MVIEW_KEYS ... 2-166
ALL_MVIEW_LOGS ... 2-166
ALL_MVIEW_REFRESH_TIMES ... 2-168
ALL_MVIEWS ... 2-168
ALL_NESTED_TABLE_COLS ... 2-173
ALL_NESTED_TABLES ... 2-175
ALL_OBJ_COLATTRS ... 2-176

ALL_OBJECT_TABLES ... 2-177
ALL_OBJECTS ... 2-181
ALL_OBJECTS_AE ... 2-183
ALL_OPANCILLARY ... 2-184
ALL_OPARGUMENTS ... 2-185
ALL_OPBINDINGS ... 2-185
ALL_OPERATOR_COMMENTS ... 2-187
ALL_OPERATORS ... 2-187
ALL_OUTLINE_HINTS ... 2-188
ALL_OUTLINES ... 2-188

## 3 Static Data Dictionary Views: ALL_PART_COL_STATISTICS to DATABASE_PROPERTIES

ALL_PART_COL_STATISTICS ... 3-1
ALL_PART_HISTOGRAMS ... 3-2
ALL_PART_INDEXES ... 3-3
ALL_PART_KEY_COLUMNS ... 3-6
ALL_PART_LOBS ... 3-7
ALL_PART_TABLES ... 3-10
ALL_PARTIAL_DROP_TABS ... 3-14
ALL_PENDING_CONV_TABLES ... 3-14
ALL_PLSQL_COLL_TYPES ... 3-15
ALL_PLSQL_OBJECT_SETTINGS ... 3-16
ALL_PLSQL_TYPE_ATTRS ... 3-17
ALL_PLSQL_TYPES ... 3-18
ALL_POLICIES ... 3-19
ALL_POLICY_ATTRIBUTES ... 3-20
ALL_POLICY_CONTEXTS ... 3-21
ALL_POLICY_GROUPS ... 3-21
ALL_PROCEDURES ... 3-22
ALL_PROPAGATION ... 3-23
ALL_QUEUE_SCHEDULES ... 3-25
ALL_QUEUE_SUBSCRIBERS ... 3-27
ALL_QUEUE_TABLES ... 3-28
ALL_QUEUES ... 3-29
ALL_REFRESH ... 3-30
ALL_REFRESH_CHILDREN ... 3-31
ALL_REFRESH_DEPENDENCIES ... 3-32
ALL_REFS ... 3-32
ALL_REGISTERED_MVIEWS ... 3-33
ALL_REGISTRY_BANNERS ... 3-34
ALL_REPL_DBNAME_MAPPING ... 3-34
ALL_REWRITE_EQUIVALENCES ... 3-35

ALL_RULE_SET_RULES ... 3-35
ALL_RULE_SETS ... 3-36
ALL_RULES ... 3-37
ALL_SCHEDULER_CHAIN_RULES ... 3-37
ALL_SCHEDULER_CHAIN_STEPS ... 3-38
ALL_SCHEDULER_CHAINS ... 3-40
ALL_SCHEDULER_CREDENTIALS ... 3-40
ALL_SCHEDULER_DB_DESTS ... 3-41
ALL_SCHEDULER_DESTS ... 3-42
ALL_SCHEDULER_EXTERNAL_DESTS ... 3-43
ALL_SCHEDULER_FILE_WATCHERS ... 3-43
ALL_SCHEDULER_GLOBAL_ATTRIBUTE ... 3-44
ALL_SCHEDULER_GROUP_MEMBERS ... 3-45
ALL_SCHEDULER_GROUPS ... 3-45
ALL_SCHEDULER_JOB_ARGS ... 3-46
ALL_SCHEDULER_JOB_CLASSES ... 3-47
ALL_SCHEDULER_JOB_DESTS ... 3-47
ALL_SCHEDULER_JOB_LOG ... 3-49
ALL_SCHEDULER_JOB_RUN_DETAILS ... 3-50
ALL_SCHEDULER_JOBS ... 3-52
ALL_SCHEDULER_NOTIFICATIONS ... 3-57
ALL_SCHEDULER_PROGRAM_ARGS ... 3-58
ALL_SCHEDULER_PROGRAMS ... 3-59
ALL_SCHEDULER_REMOTE_DATABASES ... 3-60
ALL_SCHEDULER_REMOTE_JOBSTATE ... 3-60
ALL_SCHEDULER_RUNNING_CHAINS ... 3-61
ALL_SCHEDULER_RUNNING_JOBS ... 3-63
ALL_SCHEDULER_SCHEDULES ... 3-64
ALL_SCHEDULER_WINDOW_DETAILS ... 3-65
ALL_SCHEDULER_WINDOW_GROUPS ... 3-66
ALL_SCHEDULER_WINDOW_LOG ... 3-67
ALL_SCHEDULER_WINDOWS ... 3-68
ALL_SCHEDULER_WINGROUP_MEMBERS ... 3-69
ALL_SEC_RELEVANT_COLS ... 3-69
ALL_SECONDARY_OBJECTS ... 3-70
ALL_SEQUENCES ... 3-71
ALL_SERVICES ... 3-72
ALL_SOURCE ... 3-75
ALL_SOURCE_AE ... 3-75
ALL_SQL_TRANSLATION_PROFILES ... 3-76
ALL_SQL_TRANSLATIONS ... 3-77
ALL_SQLJ_TYPE_ATTRS ... 3-78
ALL_SQLJ_TYPE_METHODS ... 3-79

ALL_SQLJ_TYPES ... 3-80
ALL_SQLSET ... 3-81
ALL_SQLSET_BINDS ... 3-82
ALL_SQLSET_PLANS ... 3-83
ALL_SQLSET_REFERENCES ... 3-88
ALL_SQLSET_STATEMENTS ... 3-88
ALL_STAT_EXTENSIONS ... 3-91
ALL_STORED_SETTINGS ... 3-91
ALL_STREAMS_COLUMNS ... 3-92
ALL_STREAMS_GLOBAL_RULES ... 3-93
ALL_STREAMS_MESSAGE_CONSUMERS ... 3-94
ALL_STREAMS_MESSAGE_RULES ... 3-95
ALL_STREAMS_NEWLY_SUPPORTED ... 3-95
ALL_STREAMS_RULES ... 3-96
ALL_STREAMS_SCHEMA_RULES ... 3-98
ALL_STREAMS_TABLE_RULES ... 3-99
ALL_STREAMS_TRANSFORM_FUNCTION ... 3-101
ALL_STREAMS_UNSUPPORTED ... 3-101
ALL_SUBPART_COL_STATISTICS ... 3-102
ALL_SUBPART_HISTOGRAMS ... 3-104
ALL_SUBPART_KEY_COLUMNS ... 3-105
ALL_SUBPARTITION_TEMPLATES ... 3-105
ALL_SUMDELTA ... 3-106
ALL_SYNC_CAPTURE ... 3-106
ALL_SYNC_CAPTURE_PREPARED_TABS ... 3-107
ALL_SYNC_CAPTURE_TABLES ... 3-108
ALL_SYNONYMS ... 3-108
ALL_TAB_COL_STATISTICS ... 3-109
ALL_TAB_COLS ... 3-111
ALL_TAB_COLUMNS ... 3-114
ALL_TAB_COMMENTS ... 3-117
ALL_TAB_HISTGRM_PENDING_STATS ... 3-118
ALL_TAB_HISTOGRAMS ... 3-119
ALL_TAB_IDENTITY_COLS ... 3-120
ALL_TAB_MODIFICATIONS ... 3-121
ALL_TAB_PARTITIONS ... 3-122
ALL_TAB_PENDING_STATS ... 3-127
ALL_TAB_PRIVS ... 3-128
ALL_TAB_PRIVS_MADE ... 3-129
ALL_TAB_PRIVS_RECD ... 3-129
ALL_TAB_STAT_PREFS ... 3-130
ALL_TAB_STATISTICS ... 3-131
ALL_TAB_STATS_HISTORY ... 3-133

ALL_TAB_SUBPARTITIONS ... 3-133
ALL_TABLES ... 3-137
ALL_TRANSFORMATIONS ... 3-142
ALL_TRIGGER_COLS ... 3-142
ALL_TRIGGER_ORDERING ... 3-143
ALL_TRIGGERS ... 3-144
ALL_TSTZ_TAB_COLS ... 3-146
ALL_TSTZ_TABLES ... 3-147
ALL_TYPE_ATTRS ... 3-147
ALL_TYPE_METHODS ... 3-148
ALL_TYPE_VERSIONS ... 3-149
ALL_TYPES ... 3-150
ALL_UNUSED_COL_TABS ... 3-151
ALL_UPDATABLE_COLUMNS ... 3-152
ALL_USERS ... 3-152
ALL_USTATS ... 3-153
ALL_VARRAYS ... 3-154
ALL_VIEWS ... 3-155
ALL_VIEWS_AE ... 3-156
ALL_WARNING_SETTINGS ... 3-157
ALL_XML_INDEXES ... 3-158
ALL_XML_NESTED_TABLES ... 3-159
ALL_XML_OUT_OF_LINE_TABLES ... 3-160
ALL_XML_SCHEMA_ATTRIBUTES ... 3-160
ALL_XML_SCHEMA_COMPLEX_TYPES ... 3-161
ALL_XML_SCHEMA_ELEMENTS ... 3-162
ALL_XML_SCHEMA_NAMESPACES ... 3-163
ALL_XML_SCHEMA_SIMPLE_TYPES ... 3-164
ALL_XML_SCHEMA_SUBSTGRP_HEAD ... 3-165
ALL_XML_SCHEMA_SUBSTGRP_MBRS ... 3-165
ALL_XML_SCHEMAS ... 3-166
ALL_XML_TAB_COLS ... 3-167
ALL_XML_TABLES ... 3-168
ALL_XML_VIEW_COLS ... 3-169
ALL_XML_VIEWS ... 3-170
ALL_XSTREAM_ADMINISTRATOR ... 3-170
ALL_XSTREAM_INBOUND ... 3-171
ALL_XSTREAM_INBOUND_PROGRESS ... 3-172
ALL_XSTREAM_OUT_SUPPORT_MODE ... 3-173
ALL_XSTREAM_OUTBOUND ... 3-174
ALL_XSTREAM_OUTBOUND_PROGRESS ... 3-175
ALL_XSTREAM_RULES ... 3-176
ALL_XSTREAM_TRANSFORMATIONS ... 3-177

ALL_ZONEMAP_MEASURES ... 3-179
ALL_ZONEMAPS ... 3-179
AUDIT_ACTIONS ... 3-181
AUDIT_UNIFIED_CONTEXTS ... 3-182
AUDIT_UNIFIED_ENABLED_POLICIES ... 3-182
AUDIT_UNIFIED_POLICIES ... 3-183
AUDIT_UNIFIED_POLICY_COMMENTS ... 3-184
AUDITABLE_SYSTEM_ACTIONS ... 3-184
CAT ... 3-186
CATALOG ... 3-186
CHAINED_ROWS ... 3-186
CLIENT_RESULT_CACHE_STATS$ ... 3-186
CLU ... 3-187
COL ... 3-187
COLS ... 3-187
DATABASE_EXPORT_OBJECTS ... 3-187
DATABASE_PROPERTIES ... 3-188

**4** Static Data Dictionary Views: DBA_2PC_NEIGHBORS to DBA_HIST_JAVA_POOL_ADVICE

DBA_2PC_NEIGHBORS ... 4-1
DBA_2PC_PENDING ... 4-1
DBA_ACL_NAME_MAP ... 4-2
DBA_ADDM_FDG_BREAKDOWN ... 4-2
DBA_ADDM_FINDINGS ... 4-3
DBA_ADDM_INSTANCES ... 4-5
DBA_ADDM_SYSTEM_DIRECTIVES ... 4-6
DBA_ADDM_TASK_DIRECTIVES ... 4-6
DBA_ADDM_TASKS ... 4-7
DBA_ADVISOR_ACTIONS ... 4-9
DBA_ADVISOR_COMMANDS ... 4-11
DBA_ADVISOR_DEF_PARAMETERS ... 4-11
DBA_ADVISOR_DEFINITIONS ... 4-12
DBA_ADVISOR_DIR_DEFINITIONS ... 4-13
DBA_ADVISOR_DIR_INSTANCES ... 4-14
DBA_ADVISOR_DIR_TASK_INST ... 4-14
DBA_ADVISOR_EXEC_PARAMETERS ... 4-15
DBA_ADVISOR_EXECUTION_TYPES ... 4-16
DBA_ADVISOR_EXECUTIONS ... 4-16
DBA_ADVISOR_FDG_BREAKDOWN ... 4-17
DBA_ADVISOR_FINDING_NAMES ... 4-18
DBA_ADVISOR_FINDINGS ... 4-18
DBA_ADVISOR_JOURNAL ... 4-19

DBA_ADVISOR_LOG ..... 4-20
DBA_ADVISOR_OBJECT_TYPES ..... 4-21
DBA_ADVISOR_OBJECTS ..... 4-22
DBA_ADVISOR_PARAMETERS ..... 4-23
DBA_ADVISOR_RATIONALE ..... 4-25
DBA_ADVISOR_RECOMMENDATIONS ..... 4-26
DBA_ADVISOR_SQLA_REC_SUM ..... 4-27
DBA_ADVISOR_SQLA_TABLES ..... 4-28
DBA_ADVISOR_SQLA_WK_MAP ..... 4-28
DBA_ADVISOR_SQLA_WK_STMTS ..... 4-29
DBA_ADVISOR_SQLPLANS ..... 4-31
DBA_ADVISOR_SQLSTATS ..... 4-34
DBA_ADVISOR_SQLW_JOURNAL ..... 4-36
DBA_ADVISOR_SQLW_PARAMETERS ..... 4-36
DBA_ADVISOR_SQLW_STMTS ..... 4-37
DBA_ADVISOR_SQLW_SUM ..... 4-39
DBA_ADVISOR_SQLW_TABLES ..... 4-40
DBA_ADVISOR_SQLW_TEMPLATES ..... 4-40
DBA_ADVISOR_TASKS ..... 4-41
DBA_ADVISOR_TEMPLATES ..... 4-43
DBA_ADVISOR_USAGE ..... 4-43
DBA_ALERT_HISTORY ..... 4-44
DBA_ALERT_HISTORY_DETAIL ..... 4-45
DBA_ALL_TABLES ..... 4-47
DBA_APPLICATION_ROLES ..... 4-47
DBA_APPLY ..... 4-47
DBA_APPLY_CHANGE_HANDLERS ..... 4-47
DBA_APPLY_CONFLICT_COLUMNS ..... 4-48
DBA_APPLY_DML_HANDLERS ..... 4-48
DBA_APPLY_ENQUEUE ..... 4-48
DBA_APPLY_ERROR ..... 4-48
DBA_APPLY_ERROR_MESSAGES ..... 4-48
DBA_APPLY_EXECUTE ..... 4-49
DBA_APPLY_INSTANTIATED_GLOBAL ..... 4-49
DBA_APPLY_INSTANTIATED_OBJECTS ..... 4-49
DBA_APPLY_INSTANTIATED_SCHEMAS ..... 4-49
DBA_APPLY_KEY_COLUMNS ..... 4-50
DBA_APPLY_OBJECT_DEPENDENCIES ..... 4-50
DBA_APPLY_PARAMETERS ..... 4-50
DBA_APPLY_PROGRESS ..... 4-50
DBA_APPLY_SPILL_TXN ..... 4-51
DBA_APPLY_TABLE_COLUMNS ..... 4-51
DBA_APPLY_VALUE_DEPENDENCIES ..... 4-51

DBA_AQ_AGENT_PRIVS ... 4-52
DBA_AQ_AGENTS ... 4-52
DBA_ARGUMENTS ... 4-52
DBA_ASSEMBLIES ... 4-53
DBA_ASSOCIATIONS ... 4-53
DBA_ATTRIBUTE_TRANSFORMATIONS ... 4-53
DBA_AUDIT_EXISTS ... 4-53
DBA_AUDIT_MGMT_CLEAN_EVENTS ... 4-55
DBA_AUDIT_MGMT_CLEANUP_JOBS ... 4-56
DBA_AUDIT_MGMT_CONFIG_PARAMS ... 4-58
DBA_AUDIT_MGMT_LAST_ARCH_TS ... 4-59
DBA_AUDIT_OBJECT ... 4-60
DBA_AUDIT_POLICIES ... 4-62
DBA_AUDIT_POLICY_COLUMNS ... 4-63
DBA_AUDIT_SESSION ... 4-63
DBA_AUDIT_STATEMENT ... 4-65
DBA_AUDIT_TRAIL ... 4-67
DBA_AUTO_SEGADV_CTL ... 4-70
DBA_AUTO_SEGADV_SUMMARY ... 4-71
DBA_AUTOTASK_CLIENT ... 4-71
DBA_AUTOTASK_CLIENT_HISTORY ... 4-73
DBA_AUTOTASK_CLIENT_JOB ... 4-73
DBA_AUTOTASK_JOB_HISTORY ... 4-74
DBA_AUTOTASK_OPERATION ... 4-75
DBA_AUTOTASK_SCHEDULE ... 4-75
DBA_AUTOTASK_STATUS ... 4-76
DBA_AUTOTASK_TASK ... 4-76
DBA_AUTOTASK_WINDOW_CLIENTS ... 4-80
DBA_AUTOTASK_WINDOW_HISTORY ... 4-80
DBA_AW_PS ... 4-81
DBA_AWS ... 4-81
DBA_BASE_TABLE_MVIEWS ... 4-81
DBA_BLOCKERS ... 4-81
DBA_CAPTURE ... 4-82
DBA_CAPTURE_EXTRA_ATTRIBUTES ... 4-82
DBA_CAPTURE_PARAMETERS ... 4-82
DBA_CAPTURE_PREPARED_DATABASE ... 4-82
DBA_CAPTURE_PREPARED_SCHEMAS ... 4-82
DBA_CAPTURE_PREPARED_TABLES ... 4-83
DBA_CATALOG ... 4-83
DBA_CDB_RSRC_PLAN_DIRECTIVES ... 4-83
DBA_CDB_RSRC_PLANS ... 4-84
DBA_CHANGE_NOTIFICATION_REGS ... 4-84

DBA_CLU_COLUMNS ... 4-85
DBA_CLUSTER_HASH_EXPRESSIONS ... 4-85
DBA_CLUSTERING_DIMENSIONS ... 4-85
DBA_CLUSTERING_JOINS ... 4-86
DBA_CLUSTERING_KEYS ... 4-86
DBA_CLUSTERING_TABLES ... 4-86
DBA_CLUSTERS ... 4-87
DBA_CODE_ROLE_PRIVS ... 4-87
DBA_COL_COMMENTS ... 4-87
DBA_COL_PENDING_STATS ... 4-87
DBA_COL_PRIVS ... 4-87
DBA_COLL_TYPES ... 4-88
DBA_COMMON_AUDIT_TRAIL ... 4-88
DBA_COMPARISON ... 4-92
DBA_COMPARISON_COLUMNS ... 4-93
DBA_COMPARISON_ROW_DIF ... 4-93
DBA_COMPARISON_SCAN ... 4-94
DBA_COMPARISON_SCAN_VALUES ... 4-95
DBA_CONNECT_ROLE_GRANTEES ... 4-95
DBA_CONS_COLUMNS ... 4-96
DBA_CONS_OBJ_COLUMNS ... 4-96
DBA_CONSTRAINTS ... 4-96
DBA_CONTAINER_DATA ... 4-96
DBA_CONTEXT ... 4-97
DBA_CPOOL_INFO ... 4-97
DBA_CPU_USAGE_STATISTICS ... 4-98
DBA_CQ_NOTIFICATION_QUERIES ... 4-99
DBA_CREDENTIALS ... 4-99
DBA_CUBE_ATTR_VISIBILITY ... 4-99
DBA_CUBE_ATTRIBUTES ... 4-99
DBA_CUBE_BUILD_PROCESSES ... 4-100
DBA_CUBE_CALCULATED_MEMBERS ... 4-100
DBA_CUBE_DIM_LEVELS ... 4-100
DBA_CUBE_DIM_MODELS ... 4-100
DBA_CUBE_DIM_VIEW_COLUMNS ... 4-100
DBA_CUBE_DIM_VIEWS ... 4-101
DBA_CUBE_DIMENSIONALITY ... 4-101
DBA_CUBE_DIMENSIONS ... 4-101
DBA_CUBE_HIER_LEVELS ... 4-101
DBA_CUBE_HIER_VIEW_COLUMNS ... 4-101
DBA_CUBE_HIER_VIEWS ... 4-102
DBA_CUBE_HIERARCHIES ... 4-102
DBA_CUBE_MEASURES ... 4-102

DBA_CUBE_NAMED_BUILD_SPECS ... 4-102
DBA_CUBE_SUB_PARTITION_LEVELS ... 4-102
DBA_CUBE_VIEW_COLUMNS ... 4-103
DBA_CUBE_VIEWS ... 4-103
DBA_CUBES ... 4-103
DBA_DATA_FILES ... 4-103
DBA_DATAPUMP_JOBS ... 4-104
DBA_DATAPUMP_SESSIONS ... 4-105
DBA_DB_LINKS ... 4-105
DBA_DBFS_HS ... 4-105
DBA_DBFS_HS_COMMANDS ... 4-106
DBA_DBFS_HS_FIXED_PROPERTIES ... 4-106
DBA_DBFS_HS_PROPERTIES ... 4-107
DBA_DDL_LOCKS ... 4-107
DBA_DEPENDENCIES ... 4-108
DBA_DIGEST_VERIFIERS ... 4-108
DBA_DIM_ATTRIBUTES ... 4-108
DBA_DIM_CHILD_OF ... 4-109
DBA_DIM_HIERARCHIES ... 4-109
DBA_DIM_JOIN_KEY ... 4-109
DBA_DIM_LEVEL_KEY ... 4-109
DBA_DIM_LEVELS ... 4-109
DBA_DIMENSIONS ... 4-109
DBA_DIRECTORIES ... 4-109
DBA_DISCOVERY_SOURCE ... 4-109
DBA_DML_LOCKS ... 4-110
DBA_DMT_FREE_SPACE ... 4-110
DBA_DMT_USED_EXTENTS ... 4-111
DBA_EDITION_COMMENTS ... 4-111
DBA_EDITIONED_TYPES ... 4-111
DBA_EDITIONING_VIEW_COLS ... 4-112
DBA_EDITIONING_VIEW_COLS_AE ... 4-112
DBA_EDITIONING_VIEWS ... 4-112
DBA_EDITIONING_VIEWS_AE ... 4-112
DBA_EDITIONS ... 4-113
DBA_ENABLED_AGGREGATIONS ... 4-113
DBA_ENABLED_TRACES ... 4-113
DBA_ENCRYPTED_COLUMNS ... 4-114
DBA_EPG_DAD_AUTHORIZATION ... 4-114
DBA_ERROR_TRANSLATIONS ... 4-114
DBA_ERRORS ... 4-115
DBA_ERRORS_AE ... 4-115
DBA_EVALUATION_CONTEXT_TABLES ... 4-115

DBA_EVALUATION_CONTEXT_VARS ..... 4-115
DBA_EVALUATION_CONTEXTS ..... 4-115
DBA_EXP_FILES ..... 4-116
DBA_EXP_OBJECTS ..... 4-116
DBA_EXP_VERSION ..... 4-116
DBA_EXTENTS ..... 4-116
DBA_EXTERNAL_LOCATIONS ..... 4-117
DBA_EXTERNAL_TABLES ..... 4-117
DBA_FEATURE_USAGE_STATISTICS ..... 4-118
DBA_FGA_AUDIT_TRAIL ..... 4-118
DBA_FILE_GROUP_EXPORT_INFO ..... 4-120
DBA_FILE_GROUP_FILES ..... 4-120
DBA_FILE_GROUP_TABLES ..... 4-120
DBA_FILE_GROUP_TABLESPACES ..... 4-121
DBA_FILE_GROUP_VERSIONS ..... 4-121
DBA_FILE_GROUPS ..... 4-121
DBA_FLASHBACK_ARCHIVE ..... 4-121
DBA_FLASHBACK_ARCHIVE_TABLES ..... 4-122
DBA_FLASHBACK_ARCHIVE_TS ..... 4-122
DBA_FLASHBACK_TXN_REPORT ..... 4-122
DBA_FLASHBACK_TXN_STATE ..... 4-123
DBA_FREE_SPACE ..... 4-124
DBA_FREE_SPACE_COALESCED ..... 4-124
DBA_GG_INBOUND_PROGRESS ..... 4-125
DBA_GLOBAL_CONTEXT ..... 4-125
DBA_GOLDENGATE_INBOUND ..... 4-126
DBA_GOLDENGATE_PRIVILEGES ..... 4-126
DBA_GOLDENGATE_RULES ..... 4-126
DBA_GOLDENGATE_SUPPORT_MODE ..... 4-126
DBA_HEAT_MAP_SEG_HISTOGRAM ..... 4-127
DBA_HEAT_MAP_SEGMENT ..... 4-127
DBA_HEATMAP_TOP_OBJECTS ..... 4-127
DBA_HEATMAP_TOP_TABLESPACES ..... 4-128
DBA_HIGH_WATER_MARK_STATISTICS ..... 4-129
DBA_HIST_ACTIVE_SESS_HISTORY ..... 4-130
DBA_HIST_APPLY_SUMMARY ..... 4-136
DBA_HIST_ASH_SNAPSHOT ..... 4-138
DBA_HIST_BASELINE ..... 4-139
DBA_HIST_BASELINE_DETAILS ..... 4-140
DBA_HIST_BASELINE_METADATA ..... 4-141
DBA_HIST_BASELINE_TEMPLATE ..... 4-142
DBA_HIST_BG_EVENT_SUMMARY ..... 4-143
DBA_HIST_BUFFER_POOL_STAT ..... 4-144

DBA_HIST_BUFFERED_QUEUES ..... 4-145
DBA_HIST_BUFFERED_SUBSCRIBERS ..... 4-147
DBA_HIST_CAPTURE ..... 4-148
DBA_HIST_CLUSTER_INTERCON ..... 4-150
DBA_HIST_COLORED_SQL ..... 4-150
DBA_HIST_COMP_IOSTAT ..... 4-151
DBA_HIST_CR_BLOCK_SERVER ..... 4-152
DBA_HIST_CURRENT_BLOCK_SERVER ..... 4-153
DBA_HIST_DATABASE_INSTANCE ..... 4-154
DBA_HIST_DATAFILE ..... 4-155
DBA_HIST_DB_CACHE_ADVICE ..... 4-156
DBA_HIST_DISPATCHER ..... 4-157
DBA_HIST_DLM_MISC ..... 4-158
DBA_HIST_DYN_REMASTER_STATS ..... 4-158
DBA_HIST_ENQUEUE_STAT ..... 4-160
DBA_HIST_EVENT_HISTOGRAM ..... 4-161
DBA_HIST_EVENT_NAME ..... 4-161
DBA_HIST_FILEMETRIC_HISTORY ..... 4-162
DBA_HIST_FILESTATXS ..... 4-163
DBA_HIST_IC_CLIENT_STATS ..... 4-164
DBA_HIST_IC_DEVICE_STATS ..... 4-165
DBA_HIST_IM_SEG_STAT ..... 4-166
DBA_HIST_IM_SEG_STAT_OBJ ..... 4-167
DBA_HIST_INST_CACHE_TRANSFER ..... 4-168
DBA_HIST_INSTANCE_RECOVERY ..... 4-170
DBA_HIST_INTERCONNECT_PINGS ..... 4-172
DBA_HIST_IOSTAT_DETAIL ..... 4-174
DBA_HIST_IOSTAT_FILETYPE ..... 4-175
DBA_HIST_IOSTAT_FILETYPE_NAME ..... 4-176
DBA_HIST_IOSTAT_FUNCTION ..... 4-177
DBA_HIST_IOSTAT_FUNCTION_NAME ..... 4-178
DBA_HIST_JAVA_POOL_ADVICE ..... 4-178

**5** Static Data Dictionary Views: DBA_HIST_LATCH to DBA_STORED_SETTINGS

DBA_HIST_LATCH ..... 5-1
DBA_HIST_LATCH_CHILDREN ..... 5-2
DBA_HIST_LATCH_MISSES_SUMMARY ..... 5-3
DBA_HIST_LATCH_NAME ..... 5-4
DBA_HIST_LATCH_PARENT ..... 5-5
DBA_HIST_LIBRARYCACHE ..... 5-6
DBA_HIST_LOG ..... 5-7
DBA_HIST_MEM_DYNAMIC_COMP ..... 5-8
DBA_HIST_MEMORY_RESIZE_OPS ..... 5-9

DBA_HIST_MEMORY_TARGET_ADVICE ..... 5-11
DBA_HIST_METRIC_NAME ..... 5-12
DBA_HIST_MTTR_TARGET_ADVICE ..... 5-12
DBA_HIST_MUTEX_SLEEP ..... 5-13
DBA_HIST_OPTIMIZER_ENV ..... 5-14
DBA_HIST_OSSTAT ..... 5-15
DBA_HIST_OSSTAT_NAME ..... 5-15
DBA_HIST_PARAMETER ..... 5-16
DBA_HIST_PARAMETER_NAME ..... 5-17
DBA_HIST_PDB_INSTANCE ..... 5-17
DBA_HIST_PERSISTENT_QMN_CACHE ..... 5-18
DBA_HIST_PERSISTENT_QUEUES ..... 5-20
DBA_HIST_PERSISTENT_SUBS ..... 5-21
DBA_HIST_PGA_TARGET_ADVICE ..... 5-23
DBA_HIST_PGASTAT ..... 5-24
DBA_HIST_PLAN_OPERATION_NAME ..... 5-25
DBA_HIST_PLAN_OPTION_NAME ..... 5-26
DBA_HIST_PROCESS_MEM_SUMMARY ..... 5-26
DBA_HIST_REPLICATION_TBL_STATS ..... 5-27
DBA_HIST_REPLICATION_TXN_STATS ..... 5-28
DBA_HIST_REPORTS ..... 5-29
DBA_HIST_REPORTS_CONTROL ..... 5-30
DBA_HIST_REPORTS_DETAILS ..... 5-31
DBA_HIST_REPORTS_TIMEBANDS ..... 5-32
DBA_HIST_RESOURCE_LIMIT ..... 5-33
DBA_HIST_ROWCACHE_SUMMARY ..... 5-34
DBA_HIST_RSRC_CONSUMER_GROUP ..... 5-35
DBA_HIST_RSRC_PLAN ..... 5-38
DBA_HIST_RULE_SET ..... 5-39
DBA_HIST_SEG_STAT ..... 5-40
DBA_HIST_SEG_STAT_OBJ ..... 5-43
DBA_HIST_SERVICE_NAME ..... 5-44
DBA_HIST_SERVICE_STAT ..... 5-44
DBA_HIST_SERVICE_WAIT_CLASS ..... 5-45
DBA_HIST_SESS_SGA_STATS ..... 5-46
DBA_HIST_SESS_TIME_STATS ..... 5-47
DBA_HIST_SESSMETRIC_HISTORY ..... 5-47
DBA_HIST_SGA ..... 5-48
DBA_HIST_SGA_TARGET_ADVICE ..... 5-49
DBA_HIST_SGASTAT ..... 5-50
DBA_HIST_SHARED_POOL_ADVICE ..... 5-50
DBA_HIST_SHARED_SERVER_SUMMARY ..... 5-52
DBA_HIST_SNAP_ERROR ..... 5-53

DBA_HIST_SNAPSHOT ..... 5-54
DBA_HIST_SQL_BIND_METADATA ..... 5-55
DBA_HIST_SQL_PLAN ..... 5-56
DBA_HIST_SQL_SUMMARY ..... 5-59
DBA_HIST_SQL_WORKAREA_HSTGRM ..... 5-59
DBA_HIST_SQLBIND ..... 5-60
DBA_HIST_SQLCOMMAND_NAME ..... 5-62
DBA_HIST_SQLSTAT ..... 5-62
DBA_HIST_SQLTEXT ..... 5-68
DBA_HIST_STAT_NAME ..... 5-68
DBA_HIST_STREAMS_APPLY_SUM ..... 5-69
DBA_HIST_STREAMS_CAPTURE ..... 5-71
DBA_HIST_STREAMS_POOL_ADVICE ..... 5-72
DBA_HIST_SYS_TIME_MODEL ..... 5-72
DBA_HIST_SYSMETRIC_HISTORY ..... 5-73
DBA_HIST_SYSMETRIC_SUMMARY ..... 5-74
DBA_HIST_SYSSTAT ..... 5-75
DBA_HIST_SYSTEM_EVENT ..... 5-75
DBA_HIST_TABLESPACE ..... 5-76
DBA_HIST_TABLESPACE_STAT ..... 5-77
DBA_HIST_TBSPC_SPACE_USAGE ..... 5-78
DBA_HIST_TEMPFILE ..... 5-79
DBA_HIST_TEMPSTATXS ..... 5-79
DBA_HIST_THREAD ..... 5-81
DBA_HIST_TOPLEVELCALL_NAME ..... 5-81
DBA_HIST_UNDOSTAT ..... 5-82
DBA_HIST_WAITCLASSMET_HISTORY ..... 5-84
DBA_HIST_WAITSTAT ..... 5-85
DBA_HIST_WR_CONTROL ..... 5-86
DBA_HISTOGRAMS ..... 5-86
DBA_HOST_ACES ..... 5-86
DBA_HOST_ACLS ..... 5-87
DBA_IDENTIFIERS ..... 5-87
DBA_ILMDATAMOVEMENTPOLICIES ..... 5-88
DBA_ILMEVALUATIONDETAILS ..... 5-89
DBA_ILMOBJECTS ..... 5-90
DBA_ILMPARAMETERS ..... 5-91
DBA_ILMPOLICIES ..... 5-92
DBA_ILMRESULTS ..... 5-93
DBA_ILMTASKS ..... 5-93
DBA_IND_COLUMNS ..... 5-94
DBA_IND_EXPRESSIONS ..... 5-94
DBA_IND_PARTITIONS ..... 5-94

DBA_IND_PENDING_STATS ..... 5-95
DBA_IND_STATISTICS ..... 5-95
DBA_IND_SUBPARTITIONS ..... 5-95
DBA_INDEXES ..... 5-95
DBA_INDEXTYPE_ARRAYTYPES ..... 5-95
DBA_INDEXTYPE_COMMENTS ..... 5-95
DBA_INDEXTYPE_OPERATORS ..... 5-96
DBA_INDEXTYPES ..... 5-96
DBA_INTERNAL_TRIGGERS ..... 5-96
DBA_JAVA_ARGUMENTS ..... 5-96
DBA_JAVA_CLASSES ..... 5-96
DBA_JAVA_COMPILER_OPTIONS ..... 5-96
DBA_JAVA_DERIVATIONS ..... 5-97
DBA_JAVA_FIELDS ..... 5-97
DBA_JAVA_IMPLEMENTS ..... 5-97
DBA_JAVA_INNERS ..... 5-97
DBA_JAVA_LAYOUTS ..... 5-97
DBA_JAVA_METHODS ..... 5-98
DBA_JAVA_NCOMPS ..... 5-98
DBA_JAVA_POLICY ..... 5-98
DBA_JAVA_RESOLVERS ..... 5-99
DBA_JAVA_THROWS ..... 5-99
DBA_JOBS ..... 5-99
DBA_JOBS_RUNNING ..... 5-100
DBA_JOIN_IND_COLUMNS ..... 5-101
DBA_JSON_COLUMNS ..... 5-101
DBA_KGLLOCK ..... 5-101
DBA_LIBRARIES ..... 5-102
DBA_LMT_FREE_SPACE ..... 5-102
DBA_LMT_USED_EXTENTS ..... 5-102
DBA_LOB_PARTITIONS ..... 5-103
DBA_LOB_SUBPARTITIONS ..... 5-103
DBA_LOB_TEMPLATES ..... 5-103
DBA_LOBS ..... 5-103
DBA_LOCK ..... 5-103
DBA_LOCK_INTERNAL ..... 5-104
DBA_LOCKS ..... 5-104
DBA_LOG_GROUP_COLUMNS ..... 5-104
DBA_LOG_GROUPS ..... 5-105
DBA_LOGMNR_LOG ..... 5-105
DBA_LOGMNR_PURGED_LOG ..... 5-106
DBA_LOGMNR_SESSION ..... 5-106
DBA_LOGSTDBY_EDS_SUPPORTED ..... 5-107

DBA_LOGSTDBY_EDS_TABLES ... 5-107
DBA_LOGSTDBY_EVENTS ... 5-107
DBA_LOGSTDBY_HISTORY ... 5-108
DBA_LOGSTDBY_LOG ... 5-110
DBA_LOGSTDBY_NOT_UNIQUE ... 5-111
DBA_LOGSTDBY_PARAMETERS ... 5-112
DBA_LOGSTDBY_PLSQL_MAP ... 5-113
DBA_LOGSTDBY_PLSQL_SUPPORT ... 5-114
DBA_LOGSTDBY_PROGRESS ... 5-114
DBA_LOGSTDBY_SKIP ... 5-114
DBA_LOGSTDBY_SKIP_TRANSACTION ... 5-115
DBA_LOGSTDBY_UNSUPPORTED ... 5-115
DBA_LOGSTDBY_UNSUPPORTED_TABLE ... 5-116
DBA_MEASURE_FOLDER_CONTENTS ... 5-117
DBA_MEASURE_FOLDER_SUBFOLDERS ... 5-117
DBA_MEASURE_FOLDERS ... 5-117
DBA_METADATA_PROPERTIES ... 5-117
DBA_METHOD_PARAMS ... 5-117
DBA_METHOD_RESULTS ... 5-118
DBA_MINING_MODEL_ATTRIBUTES ... 5-118
DBA_MINING_MODEL_SETTINGS ... 5-118
DBA_MINING_MODEL_TABLES ... 5-118
DBA_MINING_MODELS ... 5-119
DBA_MVIEW_AGGREGATES ... 5-119
DBA_MVIEW_ANALYSIS ... 5-119
DBA_MVIEW_COMMENTS ... 5-119
DBA_MVIEW_DETAIL_PARTITION ... 5-119
DBA_MVIEW_DETAIL_RELATIONS ... 5-119
DBA_MVIEW_DETAIL_SUBPARTITION ... 5-120
DBA_MVIEW_JOINS ... 5-120
DBA_MVIEW_KEYS ... 5-120
DBA_MVIEW_LOG_FILTER_COLS ... 5-120
DBA_MVIEW_LOGS ... 5-120
DBA_MVIEW_REFRESH_TIMES ... 5-120
DBA_MVIEWS ... 5-121
DBA_NESTED_TABLE_COLS ... 5-121
DBA_NESTED_TABLES ... 5-121
DBA_NETWORK_ACL_PRIVILEGES ... 5-121
DBA_NETWORK_ACLS ... 5-122
DBA_OBJ_AUDIT_OPTS ... 5-122
DBA_OBJ_COLATTRS ... 5-124
DBA_OBJECT_SIZE ... 5-124
DBA_OBJECT_TABLES ... 5-125

DBA_OBJECT_USAGE ..... 5-125
DBA_OBJECTS ..... 5-126
DBA_OBJECTS_AE ..... 5-126
DBA_OPANCILLARY ..... 5-126
DBA_OPARGUMENTS ..... 5-126
DBA_OPBINDINGS ..... 5-126
DBA_OPERATOR_COMMENTS ..... 5-127
DBA_OPERATORS ..... 5-127
DBA_OPTSTAT_OPERATION_TASKS ..... 5-127
DBA_OPTSTAT_OPERATIONS ..... 5-128
DBA_ORPHAN_KEY_TABLE ..... 5-129
DBA_OUTLINE_HINTS ..... 5-130
DBA_OUTLINES ..... 5-131
DBA_OUTSTANDING_ALERTS ..... 5-132
DBA_PARALLEL_EXECUTE_CHUNKS ..... 5-133
DBA_PARALLEL_EXECUTE_TASKS ..... 5-134
DBA_PART_COL_STATISTICS ..... 5-135
DBA_PART_HISTOGRAMS ..... 5-135
DBA_PART_INDEXES ..... 5-136
DBA_PART_KEY_COLUMNS ..... 5-136
DBA_PART_LOBS ..... 5-136
DBA_PART_TABLES ..... 5-136
DBA_PARTIAL_DROP_TABS ..... 5-137
DBA_PDB_HISTORY ..... 5-137
DBA_PDB_SAVED_STATES ..... 5-138
DBA_PDBS ..... 5-138
DBA_PENDING_CONV_TABLES ..... 5-140
DBA_PENDING_TRANSACTIONS ..... 5-140
DBA_PLSQL_COLL_TYPES ..... 5-140
DBA_PLSQL_OBJECT_SETTINGS ..... 5-140
DBA_PLSQL_TYPE_ATTRS ..... 5-140
DBA_PLSQL_TYPES ..... 5-141
DBA_POLICIES ..... 5-141
DBA_POLICY_ATTRIBUTES ..... 5-141
DBA_POLICY_CONTEXTS ..... 5-141
DBA_POLICY_GROUPS ..... 5-141
DBA_PRIV_AUDIT_OPTS ..... 5-142
DBA_PRIV_CAPTURES ..... 5-142
DBA_PROCEDURES ..... 5-143
DBA_PROFILES ..... 5-143
DBA_PROPAGATION ..... 5-144
DBA_PROXIES ..... 5-144
DBA_QUEUE_SCHEDULES ..... 5-144

DBA_QUEUE_SUBSCRIBERS ..... 5-145
DBA_QUEUE_TABLES ..... 5-145
DBA_QUEUES ..... 5-145
DBA_RCHILD ..... 5-145
DBA_RECOVERABLE_SCRIPT ..... 5-145
DBA_RECOVERABLE_SCRIPT_BLOCKS ..... 5-146
DBA_RECOVERABLE_SCRIPT_ERRORS ..... 5-146
DBA_RECOVERABLE_SCRIPT_HIST ..... 5-147
DBA_RECOVERABLE_SCRIPT_PARAMS ..... 5-147
DBA_RECYCLEBIN ..... 5-148
DBA_REDEFINITION_ERRORS ..... 5-149
DBA_REDEFINITION_OBJECTS ..... 5-150
DBA_REDEFINITION_STATUS ..... 5-150
DBA_REFRESH ..... 5-151
DBA_REFRESH_CHILDREN ..... 5-151
DBA_REFS ..... 5-151
DBA_REGISTERED_ARCHIVED_LOG ..... 5-151
DBA_REGISTERED_MVIEW_GROUPS ..... 5-152
DBA_REGISTERED_MVIEWS ..... 5-153
DBA_REGISTRY ..... 5-153
DBA_REGISTRY_HIERARCHY ..... 5-154
DBA_REGISTRY_HISTORY ..... 5-154
DBA_REGISTRY_LOG ..... 5-155
DBA_REGISTRY_SQLPATCH ..... 5-155
DBA_REPAIR_TABLE ..... 5-156
DBA_REPL_DBNAME_MAPPING ..... 5-157
DBA_RESOURCE_INCARNATIONS ..... 5-157
DBA_RESUMABLE ..... 5-158
DBA_REWRITE_EQUIVALENCES ..... 5-159
DBA_RGROUP ..... 5-159
DBA_ROLE_PRIVS ..... 5-160
DBA_ROLES ..... 5-161
DBA_ROLLBACK_SEGS ..... 5-161
DBA_ROLLING_DATABASES ..... 5-162
DBA_ROLLING_EVENTS ..... 5-163
DBA_ROLLING_PARAMETERS ..... 5-164
DBA_ROLLING_PLAN ..... 5-164
DBA_ROLLING_STATISTICS ..... 5-165
DBA_ROLLING_STATUS ..... 5-165
DBA_ROLLING_UNSUPPORTED ..... 5-166
DBA_RSRC_CATEGORIES ..... 5-167
DBA_RSRC_CONSUMER_GROUP_PRIVS ..... 5-168
DBA_RSRC_CONSUMER_GROUPS ..... 5-168

DBA_RSRC_GROUP_MAPPINGS ..... 5-169
DBA_RSRC_IO_CALIBRATE ..... 5-169
DBA_RSRC_MANAGER_SYSTEM_PRIVS ..... 5-170
DBA_RSRC_MAPPING_PRIORITY ..... 5-170
DBA_RSRC_PLAN_DIRECTIVES ..... 5-170
DBA_RSRC_PLANS ..... 5-173
DBA_RULE_SET_RULES ..... 5-174
DBA_RULE_SETS ..... 5-174
DBA_RULES ..... 5-175
DBA_SCHEDULER_CHAIN_RULES ..... 5-175
DBA_SCHEDULER_CHAIN_STEPS ..... 5-175
DBA_SCHEDULER_CHAINS ..... 5-175
DBA_SCHEDULER_CREDENTIALS ..... 5-175
DBA_SCHEDULER_DB_DESTS ..... 5-176
DBA_SCHEDULER_DESTS ..... 5-176
DBA_SCHEDULER_EXTERNAL_DESTS ..... 5-176
DBA_SCHEDULER_FILE_WATCHERS ..... 5-176
DBA_SCHEDULER_GLOBAL_ATTRIBUTE ..... 5-177
DBA_SCHEDULER_GROUP_MEMBERS ..... 5-177
DBA_SCHEDULER_GROUPS ..... 5-177
DBA_SCHEDULER_JOB_ARGS ..... 5-177
DBA_SCHEDULER_JOB_CLASSES ..... 5-177
DBA_SCHEDULER_JOB_DESTS ..... 5-178
DBA_SCHEDULER_JOB_LOG ..... 5-178
DBA_SCHEDULER_JOB_ROLES ..... 5-178
DBA_SCHEDULER_JOB_RUN_DETAILS ..... 5-180
DBA_SCHEDULER_JOBS ..... 5-180
DBA_SCHEDULER_NOTIFICATIONS ..... 5-180
DBA_SCHEDULER_PROGRAM_ARGS ..... 5-180
DBA_SCHEDULER_PROGRAMS ..... 5-180
DBA_SCHEDULER_REMOTE_DATABASES ..... 5-181
DBA_SCHEDULER_REMOTE_JOBSTATE ..... 5-181
DBA_SCHEDULER_RUNNING_CHAINS ..... 5-181
DBA_SCHEDULER_RUNNING_JOBS ..... 5-181
DBA_SCHEDULER_SCHEDULES ..... 5-181
DBA_SCHEDULER_WINDOW_DETAILS ..... 5-182
DBA_SCHEDULER_WINDOW_GROUPS ..... 5-182
DBA_SCHEDULER_WINDOW_LOG ..... 5-182
DBA_SCHEDULER_WINDOWS ..... 5-182
DBA_SCHEDULER_WINGROUP_MEMBERS ..... 5-182
DBA_SEC_RELEVANT_COLS ..... 5-183
DBA_SECONDARY_OBJECTS ..... 5-183
DBA_SEGMENTS ..... 5-183

DBA_SEGMENTS_OLD ..... 5-186
DBA_SENSITIVE_COLUMN_TYPES ..... 5-187
DBA_SENSITIVE_DATA ..... 5-187
DBA_SEQUENCES ..... 5-188
DBA_SERVER_REGISTRY ..... 5-188
DBA_SERVICES ..... 5-188
DBA_SOURCE ..... 5-189
DBA_SOURCE_AE ..... 5-189
DBA_SQL_MANAGEMENT_CONFIG ..... 5-189
DBA_SQL_PATCHES ..... 5-189
DBA_SQL_PLAN_BASELINES ..... 5-190
DBA_SQL_PLAN_DIR_OBJECTS ..... 5-192
DBA_SQL_PLAN_DIRECTIVES ..... 5-193
DBA_SQL_PROFILES ..... 5-193
DBA_SQL_TRANSLATION_PROFILES ..... 5-195
DBA_SQL_TRANSLATIONS ..... 5-195
DBA_SQLJ_TYPE_ATTRS ..... 5-195
DBA_SQLJ_TYPE_METHODS ..... 5-195
DBA_SQLJ_TYPES ..... 5-195
DBA_SQLSET ..... 5-196
DBA_SQLSET_BINDS ..... 5-196
DBA_SQLSET_PLANS ..... 5-196
DBA_SQLSET_REFERENCES ..... 5-196
DBA_SQLSET_STATEMENTS ..... 5-196
DBA_SQLTUNE_BINDS ..... 5-198
DBA_SQLTUNE_PLANS ..... 5-198
DBA_SQLTUNE_RATIONALE_PLAN ..... 5-201
DBA_SQLTUNE_STATISTICS ..... 5-202
DBA_SR_GRP_STATUS ..... 5-203
DBA_SR_GRP_STATUS_ALL ..... 5-204
DBA_SR_OBJ ..... 5-206
DBA_SR_OBJ_ALL ..... 5-207
DBA_SR_OBJ_STATUS ..... 5-207
DBA_SR_OBJ_STATUS_ALL ..... 5-208
DBA_SR_PARTN_OPS ..... 5-209
DBA_SR_STLOG_EXCEPTIONS ..... 5-210
DBA_SR_STLOG_STATS ..... 5-211
DBA_SSCR_CAPTURE ..... 5-212
DBA_SSCR_RESTORE ..... 5-212
DBA_STAT_EXTENSIONS ..... 5-213
DBA_STMT_AUDIT_OPTS ..... 5-213
DBA_STORED_SETTINGS ..... 5-214

## 6 Static Data Dictionary Views: DBA_STREAMS_ADD_COLUMN to USER_ZONEMAPS


DBA_STREAMS_ADD_COLUMN ..... 6-1
DBA_STREAMS_ADMINISTRATOR ..... 6-1
DBA_STREAMS_COLUMNS ..... 6-2
DBA_STREAMS_DELETE_COLUMN ..... 6-2
DBA_STREAMS_GLOBAL_RULES ..... 6-3
DBA_STREAMS_KEEP_COLUMNS ..... 6-3
DBA_STREAMS_MESSAGE_CONSUMERS ..... 6-3
DBA_STREAMS_MESSAGE_RULES ..... 6-4
DBA_STREAMS_NEWLY_SUPPORTED ..... 6-4
DBA_STREAMS_RENAME_COLUMN ..... 6-4
DBA_STREAMS_RENAME_SCHEMA ..... 6-5
DBA_STREAMS_RENAME_TABLE ..... 6-5
DBA_STREAMS_RULES ..... 6-5
DBA_STREAMS_SCHEMA_RULES ..... 6-6
DBA_STREAMS_SPLIT_MERGE ..... 6-6
DBA_STREAMS_SPLIT_MERGE_HIST ..... 6-8
DBA_STREAMS_STMT_HANDLERS ..... 6-10
DBA_STREAMS_STMTS ..... 6-10
DBA_STREAMS_TABLE_RULES ..... 6-10
DBA_STREAMS_TP_COMPONENT ..... 6-10
DBA_STREAMS_TP_COMPONENT_LINK ..... 6-11
DBA_STREAMS_TP_COMPONENT_STAT ..... 6-12
DBA_STREAMS_TP_DATABASE ..... 6-12
DBA_STREAMS_TP_PATH_BOTTLENECK ..... 6-13
DBA_STREAMS_TP_PATH_STAT ..... 6-14
DBA_STREAMS_TRANSFORM_FUNCTION ..... 6-14
DBA_STREAMS_TRANSFORMATIONS ..... 6-14
DBA_STREAMS_UNSUPPORTED ..... 6-15
DBA_SUBPART_COL_STATISTICS ..... 6-16
DBA_SUBPART_HISTOGRAMS ..... 6-16
DBA_SUBPART_KEY_COLUMNS ..... 6-16
DBA_SUBPARTITION_TEMPLATES ..... 6-16
DBA_SUBSCR_REGISTRATIONS ..... 6-16
DBA_SUPPLEMENTAL_LOGGING ..... 6-18
DBA_SYNC_CAPTURE ..... 6-18
DBA_SYNC_CAPTURE_PREPARED_TABS ..... 6-19
DBA_SYNC_CAPTURE_TABLES ..... 6-19
DBA_SYNONYMS ..... 6-19
DBA_SYS_PRIVS ..... 6-19
DBA_TAB_COL_STATISTICS ..... 6-20

DBA_TAB_COLS ... 6-20
DBA_TAB_COLUMNS ... 6-23
DBA_TAB_COMMENTS ... 6-26
DBA_TAB_HISTGRM_PENDING_STATS ... 6-26
DBA_TAB_HISTOGRAMS ... 6-26
DBA_TAB_IDENTITY_COLS ... 6-26
DBA_TAB_MODIFICATIONS ... 6-26
DBA_TAB_PARTITIONS ... 6-27
DBA_TAB_PENDING_STATS ... 6-27
DBA_TAB_PRIVS ... 6-27
DBA_TAB_STATISTICS ... 6-28
DBA_TAB_STAT_PREFS ... 6-28
DBA_TAB_STATS_HISTORY ... 6-28
DBA_TAB_SUBPARTITIONS ... 6-28
DBA_TABLES ... 6-28
DBA_TABLESPACE_GROUPS ... 6-29
DBA_TABLESPACE_THRESHOLDS ... 6-29
DBA_TABLESPACE_USAGE_METRICS ... 6-30
DBA_TABLESPACES ... 6-30
DBA_TEMP_FILES ... 6-33
DBA_TEMP_FREE_SPACE ... 6-33
DBA_THRESHOLDS ... 6-34
DBA_TRANSFORMATIONS ... 6-35
DBA_TRIGGER_COLS ... 6-35
DBA_TRIGGER_ORDERING ... 6-35
DBA_TRIGGERS ... 6-35
DBA_TS_QUOTAS ... 6-36
DBA_TSDP_IMPORT_ERRORS ... 6-36
DBA_TSDP_POLICY_CONDITION ... 6-37
DBA_TSDP_POLICY_FEATURE ... 6-37
DBA_TSDP_POLICY_PARAMETER ... 6-38
DBA_TSDP_POLICY_PROTECTION ... 6-38
DBA_TSDP_POLICY_TYPE ... 6-39
DBA_TSM_DESTINATION ... 6-39
DBA_TSM_SOURCE ... 6-40
DBA_TSTZ_TAB_COLS ... 6-41
DBA_TSTZ_TABLES ... 6-41
DBA_TUNE_MVIEW ... 6-41
DBA_TYPE_ATTRS ... 6-42
DBA_TYPE_METHODS ... 6-42
DBA_TYPE_VERSIONS ... 6-42
DBA_TYPES ... 6-42
DBA_UNDO_EXTENTS ... 6-42

DBA_UNUSED_COL_TABS ..... 6-43
DBA_UNUSED_OBJPRIVS ..... 6-44
DBA_UNUSED_OBJPRIVS_PATH ..... 6-44
DBA_UNUSED_PRIVS ..... 6-45
DBA_UNUSED_SYSPRIVS ..... 6-46
DBA_UNUSED_SYSPRIVS_PATH ..... 6-47
DBA_UNUSED_USERPRIVS ..... 6-48
DBA_UNUSED_USERPRIVS_PATH ..... 6-49
DBA_UPDATABLE_COLUMNS ..... 6-49
DBA_USED_OBJPRIVS ..... 6-50
DBA_USED_OBJPRIVS_PATH ..... 6-51
DBA_USED_PRIVS ..... 6-52
DBA_USED_PUBPRIVS ..... 6-53
DBA_USED_SYSPRIVS ..... 6-54
DBA_USED_SYSPRIVS_PATH ..... 6-55
DBA_USED_USERPRIVS ..... 6-55
DBA_USED_USERPRIVS_PATH ..... 6-56
DBA_USERS ..... 6-57
DBA_USERS_WITH_DEFPWD ..... 6-59
DBA_USTATS ..... 6-59
DBA_VARRAYS ..... 6-60
DBA_VIEWS ..... 6-60
DBA_VIEWS_AE ..... 6-60
DBA_WAITERS ..... 6-60
DBA_WALLET_ACES ..... 6-61
DBA_WALLET_ACLS ..... 6-61
DBA_WARNING_SETTINGS ..... 6-62
DBA_WI_CAPTURE_FILES ..... 6-62
DBA_WI_JOBS ..... 6-62
DBA_WI_OBJECTS ..... 6-63
DBA_WI_PATTERN_ITEMS ..... 6-63
DBA_WI_PATTERNS ..... 6-63
DBA_WI_STATEMENTS ..... 6-64
DBA_WI_TEMPLATE_EXECUTIONS ..... 6-64
DBA_WI_TEMPLATES ..... 6-65
DBA_WORKLOAD_ACTIVE_USER_MAP ..... 6-65
DBA_WORKLOAD_CAPTURES ..... 6-66
DBA_WORKLOAD_CONNECTION_MAP ..... 6-68
DBA_WORKLOAD_FILTERS ..... 6-69
DBA_WORKLOAD_REPLAY_DIVERGENCE ..... 6-69
DBA_WORKLOAD_REPLAY_SCHEDULES ..... 6-71
DBA_WORKLOAD_REPLAYS ..... 6-71
DBA_WORKLOAD_SCHEDULE_CAPTURES ..... 6-75

DBA_WORKLOAD_SCHEDULE_ORDERING ..... 6-76
DBA_WORKLOAD_USER_MAP ..... 6-77
DBA_XML_INDEXES ..... 6-77
DBA_XML_NESTED_TABLES ..... 6-78
DBA_XML_OUT_OF_LINE_TABLES ..... 6-78
DBA_XML_SCHEMA_ATTRIBUTES ..... 6-78
DBA_XML_SCHEMA_COMPLEX_TYPES ..... 6-78
DBA_XML_SCHEMA_ELEMENTS ..... 6-78
DBA_XML_SCHEMA_NAMESPACES ..... 6-79
DBA_XML_SCHEMA_SIMPLE_TYPES ..... 6-79
DBA_XML_SCHEMA_SUBSTGRP_HEAD ..... 6-79
DBA_XML_SCHEMA_SUBSTGRP_MBRS ..... 6-79
DBA_XML_SCHEMAS ..... 6-79
DBA_XML_TAB_COLS ..... 6-80
DBA_XML_TABLES ..... 6-80
DBA_XML_VIEW_COLS ..... 6-80
DBA_XML_VIEWS ..... 6-80
DBA_XS_AUDIT_POLICY_OPTIONS ..... 6-80
DBA_XS_AUDIT_TRAIL ..... 6-81
DBA_XS_ENB_AUDIT_POLICIES ..... 6-82
DBA_XSTREAM_ADMINISTRATOR ..... 6-83
DBA_XSTREAM_INBOUND ..... 6-83
DBA_XSTREAM_INBOUND_PROGRESS ..... 6-83
DBA_XSTREAM_OUT_SUPPORT_MODE ..... 6-83
DBA_XSTREAM_OUTBOUND ..... 6-84
DBA_XSTREAM_OUTBOUND_PROGRESS ..... 6-84
DBA_XSTREAM_RULES ..... 6-84
DBA_XSTREAM_SPLIT_MERGE ..... 6-84
DBA_XSTREAM_SPLIT_MERGE_HIST ..... 6-86
DBA_XSTREAM_STMT_HANDLERS ..... 6-88
DBA_XSTREAM_STMTS ..... 6-88
DBA_XSTREAM_TRANSFORMATIONS ..... 6-89
DBA_ZONEMAP_MEASURES ..... 6-89
DBA_ZONEMAPS ..... 6-89
DBFS_CONTENT ..... 6-90
DBFS_CONTENT_PROPERTIES ..... 6-92
DBMS_ALERT_INFO ..... 6-92
DBMS_LOCK_ALLOCATED ..... 6-92
DEPTREE ..... 6-93
DICT ..... 6-93
DICT_COLUMNS ..... 6-93
DICTIONARY ..... 6-93
DM_USER_MODELS ..... 6-94

DOCUMENT_LINKS ..... 6-95
ERROR_SIZE ..... 6-95
EXCEPTIONS ..... 6-96
FLASHBACK_TRANSACTION_QUERY ..... 6-96
GLOBAL_CONTEXT ..... 6-97
GLOBAL_NAME ..... 6-97
HS_ALL_CAPS ..... 6-97
HS_ALL_DD ..... 6-98
HS_ALL_INITS ..... 6-98
HS_BASE_CAPS ..... 6-98
HS_BASE_DD ..... 6-99
HS_CLASS_CAPS ..... 6-99
HS_CLASS_DD ..... 6-99
HS_CLASS_INIT ..... 6-100
HS_FDS_CLASS ..... 6-100
HS_FDS_INST ..... 6-100
HS_INST_CAPS ..... 6-101
HS_INST_DD ..... 6-101
HS_INST_INIT ..... 6-102
IDEPTREE ..... 6-102
IND ..... 6-102
INDEX_HISTOGRAM ..... 6-102
INDEX_STATS ..... 6-103
LOGSTDBY_UNSUPPORTED_TABLES ..... 6-104
MAP_OBJECT ..... 6-104
NLS_DATABASE_PARAMETERS ..... 6-105
NLS_INSTANCE_PARAMETERS ..... 6-105
NLS_SESSION_PARAMETERS ..... 6-106
OBJ ..... 6-106
PATH_VIEW ..... 6-106
PDB_ALERTS ..... 6-106
PDB_PLUG_IN_VIOLATIONS ..... 6-107
PLAN_TABLE ..... 6-108
PLUGGABLE_SET_CHECK ..... 6-111
PRODUCT_COMPONENT_VERSION ..... 6-112
PROXY_USERS ..... 6-112
PSTUBTBL ..... 6-113
PUBLIC_DEPENDENCY ..... 6-113
PUBLICSYN ..... 6-113
QUEUE_PRIVILEGES ..... 6-114
RECYCLEBIN ..... 6-114
REDACTION_COLUMNS ..... 6-114
REDACTION_POLICIES ..... 6-115

REDACTION_VALUES_FOR_TYPE_FULL ..... 6-116
REPORT_COMPONENTS ..... 6-117
REPORT_FILES ..... 6-117
REPORT_FORMATS ..... 6-117
RESOURCE_COST ..... 6-118
RESOURCE_MAP ..... 6-118
RESOURCE_VIEW ..... 6-119
ROLE_ROLE_PRIVS ..... 6-119
ROLE_SYS_PRIVS ..... 6-119
ROLE_TAB_PRIVS ..... 6-120
SCHEDULER_BATCH_ERRORS ..... 6-120
SCHEMA_EXPORT_OBJECTS ..... 6-121
SEQ ..... 6-121
SESSION_CONTEXT ..... 6-122
SESSION_PRIVS ..... 6-122
SESSION_ROLES ..... 6-122
SOURCE_SIZE ..... 6-122
STMT_AUDIT_OPTION_MAP ..... 6-122
SYN ..... 6-123
SYNONYMS ..... 6-123
SYS_OBJECTS ..... 6-123
SYSCATALOG ..... 6-124
SYSFILES ..... 6-124
SYSSEGOBJ ..... 6-124
SYSTEM_PRIVILEGE_MAP ..... 6-124
TAB ..... 6-124
TABLE_EXPORT_OBJECTS ..... 6-125
TABLE_PRIVILEGE_MAP ..... 6-125
TABQUOTAS ..... 6-125
TABS ..... 6-125
TRUSTED_SERVERS ..... 6-126
TS_PITR_CHECK ..... 6-126
TS_PITR_OBJECTS_TO_BE_DROPPED ..... 6-127
UNI_PLUGGABLE_SET_CHECK ..... 6-128
UNIFIED_AUDIT_TRAIL ..... 6-128
USABLE_EDITIONS ..... 6-136
USER_ADDM_FDG_BREAKDOWN ..... 6-136
USER_ADDM_FINDINGS ..... 6-136
USER_ADDM_INSTANCES ..... 6-137
USER_ADDM_TASK_DIRECTIVES ..... 6-137
USER_ADDM_TASKS ..... 6-137
USER_ADVISOR_ACTIONS ..... 6-137
USER_ADVISOR_DIR_TASK_INST ..... 6-137

USER_ADVISOR_EXEC_PARAMETERS ..... 6-137
USER_ADVISOR_EXECUTIONS ..... 6-137
USER_ADVISOR_FDG_BREAKDOWN ..... 6-138
USER_ADVISOR_FINDINGS ..... 6-138
USER_ADVISOR_JOURNAL ..... 6-138
USER_ADVISOR_LOG ..... 6-138
USER_ADVISOR_OBJECTS ..... 6-138
USER_ADVISOR_PARAMETERS ..... 6-139
USER_ADVISOR_RATIONALE ..... 6-139
USER_ADVISOR_RECOMMENDATIONS ..... 6-139
USER_ADVISOR_SQLA_REC_SUM ..... 6-139
USER_ADVISOR_SQLA_TABLES ..... 6-140
USER_ADVISOR_SQLA_WK_MAP ..... 6-140
USER_ADVISOR_SQLA_WK_STMTS ..... 6-140
USER_ADVISOR_SQLPLANS ..... 6-140
USER_ADVISOR_SQLSTATS ..... 6-140
USER_ADVISOR_SQLW_JOURNAL ..... 6-140
USER_ADVISOR_SQLW_PARAMETERS ..... 6-141
USER_ADVISOR_SQLW_STMTS ..... 6-141
USER_ADVISOR_SQLW_SUM ..... 6-141
USER_ADVISOR_SQLW_TABLES ..... 6-141
USER_ADVISOR_SQLW_TEMPLATES ..... 6-142
USER_ADVISOR_TASKS ..... 6-142
USER_ADVISOR_TEMPLATES ..... 6-142
USER_ALL_TABLES ..... 6-142
USER_APPLY_ERROR ..... 6-142
USER_AQ_AGENT_PRIVS ..... 6-143
USER_ARGUMENTS ..... 6-143
USER_ASSEMBLIES ..... 6-143
USER_ASSOCIATIONS ..... 6-143
USER_ATTRIBUTE_TRANSFORMATIONS ..... 6-144
USER_AUDIT_OBJECT ..... 6-144
USER_AUDIT_POLICIES ..... 6-144
USER_AUDIT_POLICY_COLUMNS ..... 6-145
USER_AUDIT_SESSION ..... 6-145
USER_AUDIT_STATEMENT ..... 6-145
USER_AUDIT_TRAIL ..... 6-146
USER_AW_PS ..... 6-146
USER_AWS ..... 6-146
USER_BASE_TABLE_MVIEWS ..... 6-147
USER_CATALOG ..... 6-147
USER_CHANGE_NOTIFICATION_REGS ..... 6-147
USER_CLU_COLUMNS ..... 6-147

USER_CLUSTER_HASH_EXPRESSIONS ..... 6-147
USER_CLUSTERING_DIMENSIONS ..... 6-147
USER_CLUSTERING_JOINS ..... 6-148
USER_CLUSTERING_KEYS ..... 6-148
USER_CLUSTERING_TABLES ..... 6-148
USER_CLUSTERS ..... 6-149
USER_CODE_ROLE_PRIVS ..... 6-149
USER_COL_COMMENTS ..... 6-149
USER_COL_PENDING_STATS ..... 6-149
USER_COL_PRIVS ..... 6-150
USER_COL_PRIVS_MADE ..... 6-150
USER_COL_PRIVS_RECD ..... 6-150
USER_COLL_TYPES ..... 6-150
USER_COMPARISON ..... 6-150
USER_COMPARISON_COLUMNS ..... 6-150
USER_COMPARISON_ROW_DIF ..... 6-151
USER_COMPARISON_SCAN ..... 6-151
USER_COMPARISON_SCAN_VALUES ..... 6-151
USER_CONS_COLUMNS ..... 6-151
USER_CONS_OBJ_COLUMNS ..... 6-151
USER_CONSTRAINTS ..... 6-151
USER_CQ_NOTIFICATION_QUERIES ..... 6-151
USER_CREDENTIALS ..... 6-152
USER_CUBE_ATTR_VISIBILITY ..... 6-152
USER_CUBE_ATTRIBUTES ..... 6-152
USER_CUBE_BUILD_PROCESSES ..... 6-152
USER_CUBE_CALCULATED_MEMBERS ..... 6-152
USER_CUBE_DIM_LEVELS ..... 6-153
USER_CUBE_DIM_MODELS ..... 6-153
USER_CUBE_DIM_VIEW_COLUMNS ..... 6-153
USER_CUBE_DIM_VIEWS ..... 6-153
USER_CUBE_DIMENSIONALITY ..... 6-154
USER_CUBE_DIMENSIONS ..... 6-154
USER_CUBE_HIER_LEVELS ..... 6-154
USER_CUBE_HIER_VIEW_COLUMNS ..... 6-154
USER_CUBE_HIER_VIEWS ..... 6-154
USER_CUBE_HIERARCHIES ..... 6-155
USER_CUBE_MEASURES ..... 6-155
USER_CUBE_NAMED_BUILD_SPECS ..... 6-155
USER_CUBE_SUB_PARTITION_LEVELS ..... 6-155
USER_CUBE_VIEW_COLUMNS ..... 6-156
USER_CUBE_VIEWS ..... 6-156
USER_CUBES ..... 6-156

USER_DATAPUMP_JOBS ..... 6-156
USER_DB_LINKS ..... 6-156
USER_DBFS_HS ..... 6-157
USER_DBFS_HS_COMMANDS ..... 6-157
USER_DBFS_HS_FILES ..... 6-157
USER_DBFS_HS_FIXED_PROPERTIES ..... 6-157
USER_DBFS_HS_PROPERTIES ..... 6-158
USER_DEPENDENCIES ..... 6-158
USER_DIM_ATTRIBUTES ..... 6-158
USER_DIM_CHILD_OF ..... 6-158
USER_DIM_HIERARCHIES ..... 6-158
USER_DIM_JOIN_KEY ..... 6-158
USER_DIM_LEVEL_KEY ..... 6-159
USER_DIM_LEVELS ..... 6-159
USER_DIMENSIONS ..... 6-159
USER_EDITIONED_TYPES ..... 6-159
USER_EDITIONING_VIEW_COLS ..... 6-159
USER_EDITIONING_VIEW_COLS_AE ..... 6-159
USER_EDITIONING_VIEWS ..... 6-160
USER_EDITIONING_VIEWS_AE ..... 6-160
USER_ENCRYPTED_COLUMNS ..... 6-160
USER_EPG_DAD_AUTHORIZATION ..... 6-160
USER_ERROR_TRANSLATIONS ..... 6-160
USER_ERRORS ..... 6-160
USER_ERRORS_AE ..... 6-161
USER_EVALUATION_CONTEXT_TABLES ..... 6-161
USER_EVALUATION_CONTEXT_VARS ..... 6-161
USER_EVALUATION_CONTEXTS ..... 6-161
USER_EXTENTS ..... 6-161
USER_EXTERNAL_LOCATIONS ..... 6-162
USER_EXTERNAL_TABLES ..... 6-162
USER_FILE_GROUP_EXPORT_INFO ..... 6-162
USER_FILE_GROUP_FILES ..... 6-162
USER_FILE_GROUP_TABLES ..... 6-162
USER_FILE_GROUP_TABLESPACES ..... 6-162
USER_FILE_GROUP_VERSIONS ..... 6-163
USER_FILE_GROUPS ..... 6-163
USER_FLASHBACK_ARCHIVE ..... 6-163
USER_FLASHBACK_ARCHIVE_TABLES ..... 6-163
USER_FLASHBACK_TXN_REPORT ..... 6-163
USER_FLASHBACK_TXN_STATE ..... 6-164
USER_FREE_SPACE ..... 6-164
USER_GOLDENGATE_PRIVILEGES ..... 6-164

USER_HEAT_MAP_SEG_HISTOGRAM ..... 6-164
USER_HEAT_MAP_SEGMENT ..... 6-164
USER_HISTOGRAMS ..... 6-165
USER_HOST_ACES ..... 6-165
USER_IDENTIFIERS ..... 6-165
USER_ILMDATAMOVEMENTPOLICIES ..... 6-165
USER_ILMEVALUATIONDETAILS ..... 6-166
USER_ILMOBJECTS ..... 6-166
USER_ILMPOLICIES ..... 6-166
USER_ILMRESULTS ..... 6-167
USER_ILMTASKS ..... 6-167
USER_IND_COLUMNS ..... 6-167
USER_IND_EXPRESSIONS ..... 6-167
USER_IND_PARTITIONS ..... 6-167
USER_IND_PENDING_STATS ..... 6-168
USER_IND_STATISTICS ..... 6-168
USER_IND_SUBPARTITIONS ..... 6-168
USER_INDEXES ..... 6-168
USER_INDEXTYPE_ARRAYTYPES ..... 6-168
USER_INDEXTYPE_COMMENTS ..... 6-168
USER_INDEXTYPE_OPERATORS ..... 6-169
USER_INDEXTYPES ..... 6-169
USER_INTERNAL_TRIGGERS ..... 6-169
USER_JAVA_ARGUMENTS ..... 6-169
USER_JAVA_CLASSES ..... 6-169
USER_JAVA_COMPILER_OPTIONS ..... 6-170
USER_JAVA_DERIVATIONS ..... 6-170
USER_JAVA_FIELDS ..... 6-170
USER_JAVA_IMPLEMENTS ..... 6-170
USER_JAVA_INNERS ..... 6-170
USER_JAVA_LAYOUTS ..... 6-171
USER_JAVA_METHODS ..... 6-171
USER_JAVA_NCOMPS ..... 6-171
USER_JAVA_POLICY ..... 6-171
USER_JAVA_RESOLVERS ..... 6-171
USER_JAVA_THROWS ..... 6-172
USER_JOBS ..... 6-172
USER_JOIN_IND_COLUMNS ..... 6-172
USER_JSON_COLUMNS ..... 6-172
USER_LIBRARIES ..... 6-173
USER_LOB_PARTITIONS ..... 6-173
USER_LOB_SUBPARTITIONS ..... 6-173
USER_LOB_TEMPLATES ..... 6-173

USER_LOBS ... 6-173
USER_LOG_GROUP_COLUMNS ... 6-173
USER_LOG_GROUPS ... 6-174
USER_MEASURE_FOLDER_CONTENTS ... 6-174
USER_MEASURE_FOLDER_SUBFOLDERS ... 6-174
USER_MEASURE_FOLDERS ... 6-174
USER_METADATA_PROPERTIES ... 6-174
USER_METHOD_PARAMS ... 6-175
USER_METHOD_RESULTS ... 6-175
USER_MINING_MODEL_ATTRIBUTES ... 6-175
USER_MINING_MODEL_SETTINGS ... 6-175
USER_MINING_MODELS ... 6-175
USER_MVIEW_AGGREGATES ... 6-176
USER_MVIEW_ANALYSIS ... 6-176
USER_MVIEW_COMMENTS ... 6-176
USER_MVIEW_DETAIL_PARTITION ... 6-176
USER_MVIEW_DETAIL_RELATIONS ... 6-176
USER_MVIEW_DETAIL_SUBPARTITION ... 6-177
USER_MVIEW_JOINS ... 6-177
USER_MVIEW_KEYS ... 6-177
USER_MVIEW_LOGS ... 6-177
USER_MVIEW_REFRESH_TIMES ... 6-177
USER_MVIEWS ... 6-177
USER_NESTED_TABLE_COLS ... 6-177
USER_NESTED_TABLES ... 6-178
USER_NETWORK_ACL_PRIVILEGES ... 6-178
USER_OBJ_AUDIT_OPTS ... 6-178
USER_OBJ_COLATTRS ... 6-179
USER_OBJECT_SIZE ... 6-179
USER_OBJECT_TABLES ... 6-179
USER_OBJECT_USAGE ... 6-179
USER_OBJECTS ... 6-180
USER_OBJECTS_AE ... 6-180
USER_OPANCILLARY ... 6-180
USER_OPARGUMENTS ... 6-180
USER_OPBINDINGS ... 6-180
USER_OPERATOR_COMMENTS ... 6-181
USER_OPERATORS ... 6-181
USER_OUTLINE_HINTS ... 6-181
USER_OUTLINES ... 6-181
USER_PARALLEL_EXECUTE_CHUNKS ... 6-181
USER_PARALLEL_EXECUTE_TASKS ... 6-182
USER_PART_COL_STATISTICS ... 6-182

USER_PART_HISTOGRAMS ... 6-182
USER_PART_INDEXES ... 6-182
USER_PART_KEY_COLUMNS ... 6-182
USER_PART_LOBS ... 6-183
USER_PART_TABLES ... 6-183
USER_PARTIAL_DROP_TABS ... 6-183
USER_PASSWORD_LIMITS ... 6-183
USER_PENDING_CONV_TABLES ... 6-183
USER_PLSQL_COLL_TYPES ... 6-184
USER_PLSQL_OBJECT_SETTINGS ... 6-184
USER_PLSQL_TYPE_ATTRS ... 6-184
USER_PLSQL_TYPES ... 6-184
USER_POLICIES ... 6-184
USER_POLICY_ATTRIBUTES ... 6-185
USER_POLICY_CONTEXTS ... 6-185
USER_POLICY_GROUPS ... 6-185
USER_PRIVILEGE_MAP ... 6-185
USER_PROCEDURES ... 6-186
USER_PROXIES ... 6-186
USER_QUEUE_SCHEDULES ... 6-186
USER_QUEUE_SUBSCRIBERS ... 6-186
USER_QUEUE_TABLES ... 6-186
USER_QUEUES ... 6-187
USER_RECYCLEBIN ... 6-187
USER_REFRESH ... 6-187
USER_REFRESH_CHILDREN ... 6-187
USER_REFS ... 6-187
USER_REGISTERED_MVIEWS ... 6-187
USER_REGISTRY ... 6-187
USER_RESOURCE_LIMITS ... 6-188
USER_RESUMABLE ... 6-188
USER_REWRITE_EQUIVALENCES ... 6-188
USER_ROLE_PRIVS ... 6-188
USER_RSRC_CONSUMER_GROUP_PRIVS ... 6-189
USER_RSRC_MANAGER_SYSTEM_PRIVS ... 6-189
USER_RULE_SET_RULES ... 6-189
USER_RULE_SETS ... 6-190
USER_RULES ... 6-190
USER_SCHEDULER_CHAIN_RULES ... 6-190
USER_SCHEDULER_CHAIN_STEPS ... 6-190
USER_SCHEDULER_CHAINS ... 6-190
USER_SCHEDULER_CREDENTIALS ... 6-191
USER_SCHEDULER_DB_DESTS ... 6-191

USER_SCHEDULER_DESTS ..... 6-191
USER_SCHEDULER_FILE_WATCHERS ..... 6-191
USER_SCHEDULER_GROUP_MEMBERS ..... 6-192
USER_SCHEDULER_GROUPS ..... 6-192
USER_SCHEDULER_JOB_ARGS ..... 6-192
USER_SCHEDULER_JOB_DESTS ..... 6-192
USER_SCHEDULER_JOB_LOG ..... 6-192
USER_SCHEDULER_JOB_RUN_DETAILS ..... 6-193
USER_SCHEDULER_JOBS ..... 6-193
USER_SCHEDULER_NOTIFICATIONS ..... 6-193
USER_SCHEDULER_PROGRAM_ARGS ..... 6-193
USER_SCHEDULER_PROGRAMS ..... 6-194
USER_SCHEDULER_REMOTE_JOBSTATE ..... 6-194
USER_SCHEDULER_RUNNING_CHAINS ..... 6-194
USER_SCHEDULER_RUNNING_JOBS ..... 6-194
USER_SCHEDULER_SCHEDULES ..... 6-194
USER_SEC_RELEVANT_COLS ..... 6-195
USER_SECONDARY_OBJECTS ..... 6-195
USER_SEGMENTS ..... 6-195
USER_SEQUENCES ..... 6-195
USER_SOURCE ..... 6-195
USER_SOURCE_AE ..... 6-195
USER_SQL_TRANSLATION_PROFILES ..... 6-196
USER_SQL_TRANSLATIONS ..... 6-196
USER_SQLJ_TYPE_ATTRS ..... 6-196
USER_SQLJ_TYPE_METHODS ..... 6-196
USER_SQLJ_TYPES ..... 6-196
USER_SQLSET ..... 6-197
USER_SQLSET_BINDS ..... 6-197
USER_SQLSET_PLANS ..... 6-197
USER_SQLSET_REFERENCES ..... 6-197
USER_SQLSET_STATEMENTS ..... 6-197
USER_SQLTUNE_BINDS ..... 6-197
USER_SQLTUNE_PLANS ..... 6-197
USER_SQLTUNE_RATIONALE_PLAN ..... 6-198
USER_SQLTUNE_STATISTICS ..... 6-198
USER_SR_GRP_STATUS ..... 6-198
USER_SR_GRP_STATUS_ALL ..... 6-198
USER_SR_OBJ ..... 6-198
USER_SR_OBJ_ALL ..... 6-199
USER_SR_OBJ_STATUS ..... 6-199
USER_SR_OBJ_STATUS_ALL ..... 6-199
USER_SR_PARTN_OPS ..... 6-199

USER_SR_STLOG_EXCEPTIONS ... 6-200
USER_SR_STLOG_STATS ... 6-200
USER_STAT_EXTENSIONS ... 6-200
USER_STORED_SETTINGS ... 6-200
USER_SUBPART_COL_STATISTICS ... 6-200
USER_SUBPART_HISTOGRAMS ... 6-201
USER_SUBPART_KEY_COLUMNS ... 6-201
USER_SUBPARTITION_TEMPLATES ... 6-201
USER_SUBSCR_REGISTRATIONS ... 6-201
USER_SYNONYMS ... 6-201
USER_SYS_PRIVS ... 6-201
USER_TAB_COL_STATISTICS ... 6-202
USER_TAB_COLS ... 6-202
USER_TAB_COLUMNS ... 6-202
USER_TAB_COMMENTS ... 6-203
USER_TAB_HISTGRM_PENDING_STATS ... 6-203
USER_TAB_HISTOGRAMS ... 6-203
USER_TAB_IDENTITY_COLS ... 6-203
USER_TAB_MODIFICATIONS ... 6-203
USER_TAB_PARTITIONS ... 6-204
USER_TAB_PENDING_STATS ... 6-204
USER_TAB_PRIVS ... 6-204
USER_TAB_PRIVS_MADE ... 6-204
USER_TAB_PRIVS_RECD ... 6-204
USER_TAB_STAT_PREFS ... 6-204
USER_TAB_STATISTICS ... 6-205
USER_TAB_STATS_HISTORY ... 6-205
USER_TAB_SUBPARTITIONS ... 6-205
USER_TABLES ... 6-205
USER_TABLESPACES ... 6-205
USER_TRANSFORMATIONS ... 6-205
USER_TRIGGER_COLS ... 6-206
USER_TRIGGER_ORDERING ... 6-206
USER_TRIGGERS ... 6-206
USER_TS_QUOTAS ... 6-206
USER_TSTZ_TAB_COLS ... 6-206
USER_TSTZ_TABLES ... 6-207
USER_TUNE_MVIEW ... 6-207
USER_TYPE_ATTRS ... 6-207
USER_TYPE_METHODS ... 6-207
USER_TYPE_VERSIONS ... 6-207
USER_TYPES ... 6-208
USER_UNUSED_COL_TABS ... 6-208

USER_UPDATABLE_COLUMNS ..... 6-208
USER_USERS ..... 6-208
USER_USTATS ..... 6-209
USER_VARRAYS ..... 6-210
USER_VIEWS ..... 6-210
USER_VIEWS_AE ..... 6-210
USER_WALLET_ACES ..... 6-210
USER_WARNING_SETTINGS ..... 6-210
USER_XML_INDEXES ..... 6-211
USER_XML_NESTED_TABLES ..... 6-211
USER_XML_OUT_OF_LINE_TABLES ..... 6-211
USER_XML_SCHEMA_ATTRIBUTES ..... 6-211
USER_XML_SCHEMA_COMPLEX_TYPES ..... 6-211
USER_XML_SCHEMA_ELEMENTS ..... 6-212
USER_XML_SCHEMA_NAMESPACES ..... 6-212
USER_XML_SCHEMA_SIMPLE_TYPES ..... 6-212
USER_XML_SCHEMA_SUBSTGRP_HEAD ..... 6-212
USER_XML_SCHEMA_SUBSTGRP_MBRS ..... 6-213
USER_XML_SCHEMAS ..... 6-213
USER_XML_TAB_COLS ..... 6-213
USER_XML_TABLES ..... 6-213
USER_XML_VIEW_COLS ..... 6-213
USER_XML_VIEWS ..... 6-214
USER_ZONEMAP_MEASURES ..... 6-214
USER_ZONEMAPS ..... 6-214

**Part III** Dynamic Performance Views

**7** Dynamic Performance (V$) Views: V$ACCESS to V$HVMASTER_INFO

About Dynamic Performance Views ..... 7-1
V$ Views ..... 7-2
GV$ Views ..... 7-2
Dynamic Performance View Descriptions ..... 7-3
V$ACCESS ..... 7-3
V$ACTIVE_INSTANCES ..... 7-3
V$ACTIVE_SERVICES ..... 7-4
V$ACTIVE_SESS_POOL_MTH ..... 7-6
V$ACTIVE_SESSION_HISTORY ..... 7-6
V$ADVISOR_PROGRESS ..... 7-12
V$ALERT_TYPES ..... 7-13
V$AQ ..... 7-14
V$AQ_BACKGROUND_COORDINATOR ..... 7-14
V$AQ_BMAP_NONDUR_SUBSCRIBERS ..... 7-15

V$AQ_CROSS_INSTANCE_JOBS ... 7-15
V$AQ_JOB_COORDINATOR ... 7-16
V$AQ_MESSAGE_CACHE ... 7-17
V$AQ_NONDUR_REGISTRATIONS ... 7-18
V$AQ_NONDUR_SUBSCRIBER ... 7-19
V$AQ_NONDUR_SUBSCRIBER_LWM ... 7-20
V$AQ_NOTIFICATION_CLIENTS ... 7-20
V$AQ_SERVER_POOL ... 7-21
V$AQ_SUBSCRIBER_LOAD ... 7-22
V$ARCHIVE ... 7-23
V$ARCHIVE_DEST ... 7-24
V$ARCHIVE_DEST_STATUS ... 7-27
V$ARCHIVE_GAP ... 7-29
V$ARCHIVE_PROCESSES ... 7-30
V$ARCHIVED_LOG ... 7-30
V$ASM_ACFS_ENCRYPTION_INFO ... 7-33
V$ASM_ACFS_SEC_ADMIN ... 7-34
V$ASM_ACFS_SEC_CMDRULE ... 7-35
V$ASM_ACFS_SEC_REALM ... 7-36
V$ASM_ACFS_SEC_REALM_FILTER ... 7-37
V$ASM_ACFS_SEC_REALM_GROUP ... 7-38
V$ASM_ACFS_SEC_REALM_USER ... 7-39
V$ASM_ACFS_SEC_RULE ... 7-40
V$ASM_ACFS_SEC_RULESET ... 7-41
V$ASM_ACFS_SEC_RULESET_RULE ... 7-42
V$ASM_ACFS_SECURITY_INFO ... 7-43
V$ASM_ACFSREPL ... 7-44
V$ASM_ACFSREPLTAG ... 7-46
V$ASM_ACFSSNAPSHOTS ... 7-47
V$ASM_ACFSTAG ... 7-48
V$ASM_ACFSVOLUMES ... 7-49
V$ASM_ALIAS ... 7-50
V$ASM_ATTRIBUTE ... 7-50
V$ASM_AUDIT_CLEAN_EVENTS ... 7-51
V$ASM_AUDIT_CLEANUP_JOBS ... 7-52
V$ASM_AUDIT_CONFIG_PARAMS ... 7-53
V$ASM_AUDIT_LAST_ARCH_TS ... 7-53
V$ASM_CLIENT ... 7-54
V$ASM_DISK ... 7-55
V$ASM_DISK_IOSTAT ... 7-59
V$ASM_DISK_STAT ... 7-60
V$ASM_DISKGROUP ... 7-60
V$ASM_DISKGROUP_STAT ... 7-62

V$ASM_ESTIMATE ... 7-62
V$ASM_FILE ... 7-63
V$ASM_FILESYSTEM ... 7-65
V$ASM_OPERATION ... 7-66
V$ASM_TEMPLATE ... 7-67
V$ASM_USER ... 7-68
V$ASM_USERGROUP ... 7-69
V$ASM_USERGROUP_MEMBER ... 7-69
V$ASM_VOLUME ... 7-70
V$ASM_VOLUME_STAT ... 7-71
V$AW_AGGREGATE_OP ... 7-72
V$AW_ALLOCATE_OP ... 7-72
V$AW_CALC ... 7-73
V$AW_LONGOPS ... 7-74
V$AW_OLAP ... 7-75
V$AW_SESSION_INFO ... 7-76
V$BACKUP ... 7-77
V$BACKUP_ARCHIVELOG_DETAILS ... 7-77
V$BACKUP_ARCHIVELOG_SUMMARY ... 7-78
V$BACKUP_ASYNC_IO ... 7-79
V$BACKUP_CONTROLFILE_DETAILS ... 7-81
V$BACKUP_CONTROLFILE_SUMMARY ... 7-81
V$BACKUP_COPY_DETAILS ... 7-82
V$BACKUP_COPY_SUMMARY ... 7-83
V$BACKUP_CORRUPTION ... 7-84
V$BACKUP_DATAFILE ... 7-85
V$BACKUP_DATAFILE_DETAILS ... 7-87
V$BACKUP_DATAFILE_SUMMARY ... 7-89
V$BACKUP_DEVICE ... 7-89
V$BACKUP_FILES ... 7-90
V$BACKUP_NONLOGGED ... 7-93
V$BACKUP_PIECE ... 7-94
V$BACKUP_PIECE_DETAILS ... 7-95
V$BACKUP_REDOLOG ... 7-97
V$BACKUP_SET ... 7-98
V$BACKUP_SET_DETAILS ... 7-100
V$BACKUP_SET_SUMMARY ... 7-102
V$BACKUP_SPFILE ... 7-103
V$BACKUP_SPFILE_DETAILS ... 7-103
V$BACKUP_SPFILE_SUMMARY ... 7-104
V$BACKUP_SYNC_IO ... 7-104
V$BGPROCESS ... 7-106
V$BH ... 7-106

V$BLOCK_CHANGE_TRACKING ... 7-108
V$BLOCKING_QUIESCE ... 7-108
V$BT_SCAN_CACHE ... 7-109
V$BT_SCAN_OBJ_TEMPS ... 7-109
V$BUFFER_POOL ... 7-110
V$BUFFER_POOL_STATISTICS ... 7-111
V$BUFFERED_PUBLISHERS ... 7-113
V$BUFFERED_QUEUES ... 7-114
V$BUFFERED_SUBSCRIBERS ... 7-115
V$CACHE ... 7-116
V$CACHE_LOCK ... 7-118
V$CACHE_TRANSFER ... 7-119
V$CIRCUIT ... 7-121
V$CLASS_CACHE_TRANSFER ... 7-122
V$CLIENT_SECRETS ... 7-123
V$CLIENT_STATS ... 7-124
V$CLONEDFILE ... 7-124
V$CLUSTER_INTERCONNECTS ... 7-125
V$CON_SYS_TIME_MODEL ... 7-126
V$CON_SYSSTAT ... 7-126
V$CON_SYSTEM_EVENT ... 7-127
V$CON_SYSTEM_WAIT_CLASS ... 7-128
V$CONFIGURED_INTERCONNECTS ... 7-129
V$CONTAINERS ... 7-129
V$CONTEXT ... 7-130
V$CONTROLFILE ... 7-131
V$CONTROLFILE_RECORD_SECTION ... 7-131
V$COPY_CORRUPTION ... 7-132
V$COPY_NONLOGGED ... 7-133
V$CORRUPT_XID_LIST ... 7-134
V$CPOOL_CC_INFO ... 7-134
V$CPOOL_CC_STATS ... 7-135
V$CPOOL_CONN_INFO ... 7-135
V$CPOOL_STATS ... 7-137
V$CR_BLOCK_SERVER ... 7-137
V$CURRENT_BLOCK_SERVER ... 7-139
V$DATABASE ... 7-140
V$DATABASE_BLOCK_CORRUPTION ... 7-147
V$DATABASE_INCARNATION ... 7-148
V$DATAFILE ... 7-149
V$DATAFILE_COPY ... 7-151
V$DATAFILE_HEADER ... 7-154
V$DATAGUARD_CONFIG ... 7-155

V$DATAGUARD_STATS ... 7-156
V$DATAGUARD_STATUS ... 7-157
V$DB_CACHE_ADVICE ... 7-158
V$DB_OBJECT_CACHE ... 7-159
V$DB_PIPES ... 7-161
V$DB_TRANSPORTABLE_PLATFORM ... 7-161
V$DBFILE ... 7-162
V$DBLINK ... 7-162
V$DEAD_CLEANUP ... 7-163
V$DELETED_OBJECT ... 7-164
V$DG_BROKER_CONFIG ... 7-165
V$DIAG_INFO ... 7-166
V$DISPATCHER ... 7-166
V$DISPATCHER_CONFIG ... 7-167
V$DISPATCHER_RATE ... 7-168
V$DNFS_CHANNELS ... 7-173
V$DNFS_FILES ... 7-174
V$DNFS_SERVERS ... 7-175
V$DNFS_STATS ... 7-175
V$DYNAMIC_REMASTER_STATS ... 7-177
V$EDITIONABLE_TYPES ... 7-178
V$EMON ... 7-178
V$EMX_USAGE_STATS ... 7-180
V$ENABLEDPRIVS ... 7-181
V$ENCRYPTED_TABLESPACES ... 7-181
V$ENCRYPTION_KEYS ... 7-182
V$ENCRYPTION_WALLET ... 7-184
V$ENQUEUE_LOCK ... 7-185
V$ENQUEUE_STAT ... 7-186
V$ENQUEUE_STATISTICS ... 7-186
V$EVENT_HISTOGRAM ... 7-187
V$EVENT_HISTOGRAM_MICRO ... 7-188
V$EVENT_NAME ... 7-189
V$EVENTMETRIC ... 7-189
V$EXECUTION ... 7-190
V$FALSE_PING ... 7-191
V$FAST_START_SERVERS ... 7-192
V$FAST_START_TRANSACTIONS ... 7-192
V$FILE_CACHE_TRANSFER ... 7-193
V$FILE_HISTOGRAM ... 7-194
V$FILEMETRIC ... 7-195
V$FILEMETRIC_HISTORY ... 7-195
V$FILESPACE_USAGE ... 7-196

V$FILESTAT ..... 7-196
V$FIXED_TABLE ..... 7-197
V$FIXED_VIEW_DEFINITION ..... 7-198
V$FLASHBACK_DATABASE_LOG ..... 7-198
V$FLASHBACK_DATABASE_LOGFILE ..... 7-199
V$FLASHBACK_DATABASE_STAT ..... 7-200
V$FLASHBACK_TXN_GRAPH ..... 7-200
V$FLASHBACK_TXN_MODS ..... 7-201
V$FLASHFILESTAT ..... 7-203
V$FOREIGN_ARCHIVED_LOG ..... 7-203
V$FS_FAILOVER_STATS ..... 7-206
V$FS_OBSERVER_HISTOGRAM ..... 7-206
V$GC_ELEMENT ..... 7-207
V$GC_ELEMENTS_WITH_COLLISIONS ..... 7-208
V$GCSHVMASTER_INFO ..... 7-208
V$GCSPFMASTER_INFO ..... 7-209
V$GES_BLOCKING_ENQUEUE ..... 7-209
V$GES_CONVERT_LOCAL ..... 7-211
V$GES_CONVERT_REMOTE ..... 7-211
V$GES_ENQUEUE ..... 7-212
V$GES_LATCH ..... 7-214
V$GES_RESOURCE ..... 7-214
V$GES_STATISTICS ..... 7-214
V$GG_APPLY_COORDINATOR ..... 7-215
V$GG_APPLY_READER ..... 7-217
V$GG_APPLY_RECEIVER ..... 7-219
V$GG_APPLY_SERVER ..... 7-220
V$GLOBAL_BLOCKED_LOCKS ..... 7-223
V$GLOBAL_TRANSACTION ..... 7-224
V$GOLDENGATE_CAPTURE ..... 7-225
V$GOLDENGATE_MESSAGE_TRACKING ..... 7-228
V$GOLDENGATE_TABLE_STATS ..... 7-230
V$GOLDENGATE_TRANSACTION ..... 7-231
V$HANG_INFO ..... 7-233
V$HANG_SESSION_INFO ..... 7-235
V$HANG_STATISTICS ..... 7-236
V$HEAT_MAP_SEGMENT ..... 7-236
V$HM_CHECK ..... 7-237
V$HM_CHECK_PARAM ..... 7-238
V$HM_FINDING ..... 7-238
V$HM_INFO ..... 7-239
V$HM_RECOMMENDATION ..... 7-240
V$HM_RUN ..... 7-241

V$HS_AGENT ..... 7-242
V$HS_PARAMETER ..... 7-242
V$HS_SESSION ..... 7-243
V$HVMASTER_INFO ..... 7-243

## 8 Dynamic Performance (V$) Views: V$IM_COLUMN_LEVEL to V$RULE_SET_AGGREGATE_STATS

V$IM_COLUMN_LEVEL ..... 8-1
V$IM_SEGMENTS ..... 8-2
V$IM_USER_SEGMENTS ..... 8-3
V$INDEXED_FIXED_COLUMN ..... 8-5
V$INMEMORY_AREA ..... 8-6
V$INSTANCE ..... 8-6
V$INSTANCE_CACHE_TRANSFER ..... 8-8
V$INSTANCE_PING ..... 8-10
V$INSTANCE_RECOVERY ..... 8-11
V$IO_CALIBRATION_STATUS ..... 8-13
V$IO_OUTLIER ..... 8-13
V$IOFUNCMETRIC ..... 8-14
V$IOFUNCMETRIC_HISTORY ..... 8-15
V$IOSTAT_CONSUMER_GROUP ..... 8-15
V$IOSTAT_FILE ..... 8-16
V$IOSTAT_FUNCTION ..... 8-17
V$IOSTAT_FUNCTION_DETAIL ..... 8-18
V$IOSTAT_NETWORK ..... 8-19
V$JAVA_LIBRARY_CACHE_MEMORY ..... 8-20
V$JAVA_POOL_ADVICE ..... 8-21
V$KERNEL_IO_OUTLIER ..... 8-21
V$KEY_VECTOR ..... 8-22
V$LATCH ..... 8-25
V$LATCH_CHILDREN ..... 8-26
V$LATCH_MISSES ..... 8-27
V$LATCH_PARENT ..... 8-28
V$LATCHHOLDER ..... 8-28
V$LATCHNAME ..... 8-28
V$LGWRIO_OUTLIER ..... 8-29
V$LIBCACHE_LOCKS ..... 8-30
V$LIBRARY_CACHE_MEMORY ..... 8-31
V$LIBRARYCACHE ..... 8-31
V$LICENSE ..... 8-32
V$LOADISTAT ..... 8-33
V$LOADPSTAT ..... 8-33
V$LOCK ..... 8-34

V$LOCK_ACTIVITY ..... 8-36
V$LOCK_TYPE ..... 8-37
V$LOCKED_OBJECT ..... 8-37
V$LOG ..... 8-38
V$LOG_HISTORY ..... 8-39
V$LOGFILE ..... 8-40
V$LOGHIST ..... 8-40
V$LOGMNR_CONTENTS ..... 8-41
V$LOGMNR_DICTIONARY ..... 8-47
V$LOGMNR_DICTIONARY_LOAD ..... 8-48
V$LOGMNR_LATCH ..... 8-48
V$LOGMNR_LOGS ..... 8-49
V$LOGMNR_PARAMETERS ..... 8-50
V$LOGMNR_PROCESS ..... 8-51
V$LOGMNR_SESSION ..... 8-52
V$LOGMNR_STATS ..... 8-54
V$LOGSTDBY ..... 8-55
V$LOGSTDBY_PROCESS ..... 8-56
V$LOGSTDBY_PROGRESS ..... 8-57
V$LOGSTDBY_STATE ..... 8-58
V$LOGSTDBY_STATS ..... 8-59
V$LOGSTDBY_TRANSACTION ..... 8-60
V$MANAGED_STANDBY ..... 8-62
V$MAP_COMP_LIST ..... 8-63
V$MAP_ELEMENT ..... 8-64
V$MAP_EXT_ELEMENT ..... 8-65
V$MAP_FILE ..... 8-65
V$MAP_FILE_EXTENT ..... 8-66
V$MAP_FILE_IO_STACK ..... 8-67
V$MAP_LIBRARY ..... 8-68
V$MAP_SUBELEMENT ..... 8-68
V$MAPPED_SQL ..... 8-69
V$MEMORY_CURRENT_RESIZE_OPS ..... 8-70
V$MEMORY_DYNAMIC_COMPONENTS ..... 8-70
V$MEMORY_RESIZE_OPS ..... 8-71
V$MEMORY_TARGET_ADVICE ..... 8-72
V$METRIC ..... 8-73
V$METRICGROUP ..... 8-74
V$METRIC_HISTORY ..... 8-75
V$METRICNAME ..... 8-76
V$MTTR_TARGET_ADVICE ..... 8-76
V$MUTEX_SLEEP ..... 8-77
V$MUTEX_SLEEP_HISTORY ..... 8-78

V$MVREFRESH ... 8-79
V$MYSTAT ... 8-79
V$NFS_CLIENTS ... 8-79
V$NFS_LOCKS ... 8-80
V$NFS_OPEN_FILES ... 8-81
V$NLS_PARAMETERS ... 8-81
V$NLS_VALID_VALUES ... 8-82
V$NONLOGGED_BLOCK ... 8-82
V$OBJECT_DEPENDENCY ... 8-83
V$OBJECT_PRIVILEGE ... 8-84
V$OBJECT_USAGE ... 8-84
V$OBSOLETE_BACKUP_FILES ... 8-85
V$OBSOLETE_PARAMETER ... 8-87
V$OFFLINE_RANGE ... 8-88
V$OPEN_CURSOR ... 8-88
V$OPTIMIZER_PROCESSING_RATE ... 8-89
V$OPTION ... 8-90
V$OSSTAT ... 8-91
V$PARALLEL_DEGREE_LIMIT_MTH ... 8-93
V$PARAMETER ... 8-93
V$PARAMETER_VALID_VALUES ... 8-95
V$PARAMETER2 ... 8-95
V$PATCHES ... 8-97
V$PDB_INCARNATION ... 8-98
V$PDBS ... 8-99
V$PERSISTENT_PUBLISHERS ... 8-100
V$PERSISTENT_QMN_CACHE ... 8-100
V$PERSISTENT_QUEUES ... 8-102
V$PERSISTENT_SUBSCRIBERS ... 8-103
V$PGA_TARGET_ADVICE ... 8-105
V$PGA_TARGET_ADVICE_HISTOGRAM ... 8-106
V$PGASTAT ... 8-108
V$PQ_SESSTAT ... 8-110
V$PQ_SLAVE ... 8-111
V$PQ_SYSSTAT ... 8-112
V$PQ_TQSTAT ... 8-112
V$PROCESS ... 8-113
V$PROCESS_MEMORY ... 8-115
V$PROPAGATION_RECEIVER ... 8-115
V$PROPAGATION_SENDER ... 8-117
V$PROXY_ARCHIVEDLOG ... 8-118
V$PROXY_ARCHIVELOG_DETAILS ... 8-120
V$PROXY_ARCHIVELOG_SUMMARY ... 8-121

V$PROXY_COPY_DETAILS ..... 8-122
V$PROXY_COPY_SUMMARY ..... 8-123
V$PROXY_DATAFILE ..... 8-124
V$PWFILE_USERS ..... 8-127
V$PX_INSTANCE_GROUP ..... 8-127
V$PX_PROCESS ..... 8-128
V$PX_PROCESS_SYSSTAT ..... 8-128
V$PX_SESSION ..... 8-130
V$PX_SESSTAT ..... 8-130
V$QMON_COORDINATOR_STATS ..... 8-131
V$QMON_SERVER_STATS ..... 8-132
V$QMON_TASK_STATS ..... 8-133
V$QMON_TASKS ..... 8-134
V$QUEUE ..... 8-135
V$QUEUEING_MTH ..... 8-135
V$RECOVER_FILE ..... 8-135
V$RECOVERY_AREA_USAGE ..... 8-136
V$RECOVERY_FILE_DEST ..... 8-137
V$RECOVERY_FILE_STATUS ..... 8-137
V$RECOVERY_LOG ..... 8-138
V$RECOVERY_PROGRESS ..... 8-139
V$RECOVERY_STATUS ..... 8-140
V$REDO_DEST_RESP_HISTOGRAM ..... 8-141
V$REPLPROP ..... 8-141
V$REPLQUEUE ..... 8-143
V$REQDIST ..... 8-143
V$RESERVED_WORDS ..... 8-144
V$RESOURCE ..... 8-144
V$RESOURCE_LIMIT ..... 8-145
V$RESTORE_POINT ..... 8-146
V$RESULT_CACHE_DEPENDENCY ..... 8-147
V$RESULT_CACHE_MEMORY ..... 8-147
V$RESULT_CACHE_OBJECTS ..... 8-148
V$RESULT_CACHE_STATISTICS ..... 8-149
V$RMAN_BACKUP_JOB_DETAILS ..... 8-150
V$RMAN_BACKUP_SUBJOB_DETAILS ..... 8-152
V$RMAN_BACKUP_TYPE ..... 8-153
V$RMAN_COMPRESSION_ALGORITHM ..... 8-154
V$RMAN_CONFIGURATION ..... 8-155
V$RMAN_ENCRYPTION_ALGORITHMS ..... 8-155
V$RMAN_OUTPUT ..... 8-156
V$RMAN_STATUS ..... 8-156
V$RO_USER_ACCOUNT ..... 8-158

V$ROLLNAME ... 8-159
V$ROLLSTAT ... 8-159
V$ROWCACHE ... 8-160
V$ROWCACHE_PARENT ... 8-161
V$ROWCACHE_SUBORDINATE ... 8-162
V$RSRC_CONS_GROUP_HISTORY ... 8-163
V$RSRC_CONSUMER_GROUP ... 8-166
V$RSRC_CONSUMER_GROUP_CPU_MTH ... 8-169
V$RSRC_PLAN ... 8-170
V$RSRC_PLAN_CPU_MTH ... 8-171
V$RSRC_PLAN_HISTORY ... 8-172
V$RSRC_SESSION_INFO ... 8-173
V$RSRCMGRMETRIC ... 8-178
V$RSRCMGRMETRIC_HISTORY ... 8-180
V$RULE ... 8-181
V$RULE_SET ... 8-181
V$RULE_SET_AGGREGATE_STATS ... 8-183

**9** Dynamic Performance (V$) Views: V$SCHEDULER_RUNNING_JOBS to V$XSTREAM_TRANSACTION

V$SCHEDULER_RUNNING_JOBS ... 9-1
V$SECUREFILE_TIMER ... 9-1
V$SEGMENT_STATISTICS ... 9-2
V$SEGSTAT ... 9-3
V$SEGSTAT_NAME ... 9-3
V$SERV_MOD_ACT_STATS ... 9-4
V$SERVICE_EVENT ... 9-4
V$SERVICE_STATS ... 9-5
V$SERVICE_WAIT_CLASS ... 9-6
V$SERVICEMETRIC ... 9-6
V$SERVICEMETRIC_HISTORY ... 9-7
V$SERVICES ... 9-8
V$SES_OPTIMIZER_ENV ... 9-10
V$SESS_IO ... 9-10
V$SESS_TIME_MODEL ... 9-11
V$SESSION ... 9-14
V$SESSION_BLOCKERS ... 9-20
V$SESSION_CONNECT_INFO ... 9-21
V$SESSION_CURSOR_CACHE ... 9-22
V$SESSION_EVENT ... 9-23
V$SESSION_FIX_CONTROL ... 9-24
V$SESSION_LONGOPS ... 9-25
V$SESSION_OBJECT_CACHE ... 9-27

V$SESSION_WAIT ..... 9-27
V$SESSION_WAIT_CLASS ..... 9-29
V$SESSION_WAIT_HISTORY ..... 9-30
V$SESSIONS_COUNT ..... 9-30
V$SESSMETRIC ..... 9-31
V$SESSTAT ..... 9-31
V$SGA ..... 9-32
V$SGA_CURRENT_RESIZE_OPS ..... 9-32
V$SGA_DYNAMIC_COMPONENTS ..... 9-33
V$SGA_DYNAMIC_FREE_MEMORY ..... 9-34
V$SGA_RESIZE_OPS ..... 9-34
V$SGA_TARGET_ADVICE ..... 9-35
V$SGAINFO ..... 9-36
V$SGASTAT ..... 9-36
V$SHARED_POOL_ADVICE ..... 9-37
V$SHARED_POOL_RESERVED ..... 9-38
V$SHARED_SERVER ..... 9-39
V$SHARED_SERVER_MONITOR ..... 9-40
V$SORT_SEGMENT ..... 9-40
V$SPPARAMETER ..... 9-41
V$SQL ..... 9-42
V$SQL_BIND_CAPTURE ..... 9-47
V$SQL_BIND_DATA ..... 9-49
V$SQL_BIND_METADATA ..... 9-50
V$SQL_CS_HISTOGRAM ..... 9-51
V$SQL_CS_SELECTIVITY ..... 9-51
V$SQL_CS_STATISTICS ..... 9-52
V$SQL_CURSOR ..... 9-53
V$SQL_JOIN_FILTER ..... 9-54
V$SQL_MONITOR ..... 9-54
V$SQL_MONITOR_SESSTAT ..... 9-59
V$SQL_MONITOR_STATNAME ..... 9-60
V$SQL_OPTIMIZER_ENV ..... 9-60
V$SQL_PLAN ..... 9-61
V$SQL_PLAN_MONITOR ..... 9-64
V$SQL_PLAN_STATISTICS ..... 9-69
V$SQL_PLAN_STATISTICS_ALL ..... 9-70
V$SQL_REDIRECTION ..... 9-75
V$SQL_SHARED_CURSOR ..... 9-76
V$SQL_SHARED_MEMORY ..... 9-79
V$SQL_WORKAREA ..... 9-80
V$SQL_WORKAREA_ACTIVE ..... 9-82
V$SQL_WORKAREA_HISTOGRAM ..... 9-84

V$SQLAREA ..... 9-85
V$SQLAREA_PLAN_HASH ..... 9-90
V$SQLCOMMAND ..... 9-94
V$SQLFN_ARG_METADATA ..... 9-95
V$SQLFN_METADATA ..... 9-95
V$SQLSTATS ..... 9-96
V$SQLSTATS_PLAN_HASH ..... 9-101
V$SQLTEXT ..... 9-101
V$SQLTEXT_WITH_NEWLINES ..... 9-101
V$STANDBY_EVENT_HISTOGRAM ..... 9-102
V$STANDBY_LOG ..... 9-103
V$STATISTICS_LEVEL ..... 9-104
V$STATNAME ..... 9-105
V$STREAMS_APPLY_COORDINATOR ..... 9-106
V$STREAMS_APPLY_READER ..... 9-108
V$STREAMS_APPLY_SERVER ..... 9-111
V$STREAMS_CAPTURE ..... 9-114
V$STREAMS_MESSAGE_TRACKING ..... 9-117
V$STREAMS_POOL_ADVICE ..... 9-118
V$STREAMS_POOL_STATISTICS ..... 9-119
V$STREAMS_TRANSACTION ..... 9-120
V$SUBCACHE ..... 9-121
V$SUBSCR_REGISTRATION_STATS ..... 9-122
V$SYS_OPTIMIZER_ENV ..... 9-123
V$SYS_TIME_MODEL ..... 9-124
V$SYSAUX_OCCUPANTS ..... 9-124
V$SYSMETRIC ..... 9-125
V$SYSMETRIC_HISTORY ..... 9-125
V$SYSMETRIC_SUMMARY ..... 9-126
V$SYSSTAT ..... 9-127
V$SYSTEM_CURSOR_CACHE ..... 9-128
V$SYSTEM_EVENT ..... 9-128
V$SYSTEM_FIX_CONTROL ..... 9-129
V$SYSTEM_PARAMETER ..... 9-130
V$SYSTEM_PARAMETER2 ..... 9-131
V$SYSTEM_WAIT_CLASS ..... 9-133
V$TABLESPACE ..... 9-134
V$TEMP_CACHE_TRANSFER ..... 9-134
V$TEMP_EXTENT_MAP ..... 9-135
V$TEMP_EXTENT_POOL ..... 9-136
V$TEMP_SPACE_HEADER ..... 9-137
V$TEMPFILE ..... 9-137
V$TEMPORARY_LOBS ..... 9-138

V$TEMPSEG_USAGE ... 9-138
V$TEMPSTAT ... 9-139
V$TEMPUNDOSTAT ... 9-140
V$THREAD ... 9-141
V$THRESHOLD_TYPES ... 9-142
V$TIMER ... 9-143
V$TIMEZONE_NAMES ... 9-143
V$TOPLEVELCALL ... 9-143
V$TRANSACTION ... 9-144
V$TRANSACTION_ENQUEUE ... 9-146
V$TRANSPORTABLE_PLATFORM ... 9-146
V$TSDP_SUPPORTED_FEATURE ... 9-147
V$TYPE_SIZE ... 9-147
V$UNDOSTAT ... 9-148
V$UNUSABLE_BACKUPFILE_DETAILS ... 9-150
V$VERSION ... 9-151
V$VPD_POLICY ... 9-151
V$WAIT_CHAINS ... 9-152
V$WAITCLASSMETRIC ... 9-154
V$WAITCLASSMETRIC_HISTORY ... 9-155
V$WAITSTAT ... 9-155
V$WALLET ... 9-155
V$WORKLOAD_REPLAY_THREAD ... 9-156
V$XML_AUDIT_TRAIL ... 9-158
V$XSTREAM_APPLY_COORDINATOR ... 9-160
V$XSTREAM_APPLY_READER ... 9-163
V$XSTREAM_APPLY_RECEIVER ... 9-165
V$XSTREAM_APPLY_SERVER ... 9-166
V$XSTREAM_CAPTURE ... 9-169
V$XSTREAM_MESSAGE_TRACKING ... 9-173
V$XSTREAM_OUTBOUND_SERVER ... 9-174
V$XSTREAM_TABLE_STATS ... 9-177
V$XSTREAM_TRANSACTION ... 9-179

**Part IV** Appendixes

**A** Database Limits

Datatype Limits ... A-1
Physical Database Limits ... A-3
Logical Database Limits ... A-4
Process and Runtime Limits ... A-5

## B SQL Scripts


Creating the Data Dictionary ..... B-1
Creating Additional Data Dictionary Structures ..... B-2
The "NO" Scripts ..... B-5
Upgrade Scripts ..... B-6
Java Scripts ..... B-6


## C Oracle Wait Events


Classes of Wait Events ..... C-1
Descriptions of Common Wait Event Parameters ..... C-2
Descriptions of Wait Events ..... C-5
- alter system set dispatcher ..... C-5
- ARCH Remote Write ..... C-5
- ASYNC Remote Write ..... C-5
- batched allocate scn lock request ..... C-5
- BFILE check if exists ..... C-6
- BFILE check if open ..... C-6
- BFILE closure ..... C-6
- BFILE get length ..... C-6
- BFILE get name object ..... C-6
- BFILE get path object ..... C-7
- BFILE internal seek ..... C-7
- BFILE open ..... C-7
- BFILE read ..... C-7
- broadcast mesg queue transition ..... C-7
- broadcast mesg recovery queue transition ..... C-8
- buffer busy waits ..... C-8
- buffer deadlock ..... C-9
- buffer latch ..... C-9
- buffer read retry ..... C-10
- checkpoint completed ..... C-10
- cleanup of aborted processes ..... C-10
- control file parallel write ..... C-10
- control file sequential read ..... C-11
- control file single write ..... C-11
- cursor: mutex S ..... C-11
- cursor: mutex X ..... C-12
- cursor: pin S ..... C-12
- cursor: pin S wait on X ..... C-12
- cursor: pin X ..... C-13
- Data Guard: process clean up ..... C-13
- Data Guard: process exit ..... C-13

Data Guard Broker: single instance ..... C-13
db file asynch I/O submit ..... C-13
db file parallel read ..... C-13
db file parallel write ..... C-14
db file scattered read ..... C-14
db file sequential read ..... C-15
db file single write ..... C-15
DFS db file lock ..... C-16
DFS lock handle ..... C-16
direct path read ..... C-16
direct path sync ..... C-17
direct path write ..... C-17
Disk file operations I/O ..... C-18
dispatcher shutdown ..... C-18
dispatcher timer ..... C-18
duplicate cluster key ..... C-19
enq: OW - initialization ..... C-19
enq: OW - termination ..... C-19
enqueue ..... C-19
flashback buf free by RVWR ..... C-20
flashback logfile sync ..... C-20
free buffer waits ..... C-20
free global transaction table entry ..... C-21
free process state object ..... C-21
gc recovery quiesce ..... C-21
GCS lock open S ..... C-21
GCS lock open X ..... C-22
gcs remastering wait for drop pkey ..... C-22
global cache busy ..... C-22
global cache lock cleanup ..... C-22
global cache freelist ..... C-23
inactive session ..... C-23
inactive transaction branch ..... C-23
index block split ..... C-23
instance state change ..... C-24
io done ..... C-24
kcl bg acks ..... C-24
ksxr wait for mount shared ..... C-25
ktm: instance recovery ..... C-25
latch activity ..... C-25
latch free ..... C-26
library cache load lock ..... C-26
library cache lock ..... C-26

library cache pin ..... C-27
library cache shutdown ..... C-27
LMON global data update ..... C-27
lock manager wait for remote message ..... C-28
Log archive I/O ..... C-28
log buffer space ..... C-28
log file parallel write ..... C-28
log file sequential read ..... C-29
log file single write ..... C-29
log file switch (archiving needed) ..... C-29
log file switch (checkpoint incomplete) ..... C-29
log file switch (clearing log file) ..... C-30
log file switch (private strand flush incomplete) ..... C-30
log file switch completion ..... C-30
log file sync ..... C-30
log switch/archive ..... C-30
optimizer stats update retry ..... C-31
parallel recovery change buffer free ..... C-31
parallel recovery control message reply ..... C-31
parallel recovery coord send blocked ..... C-31
parallel recovery coord wait for reply ..... C-31
parallel recovery coordinator waits for slave cleanup ..... C-31
parallel recovery read buffer free ..... C-32
parallel recovery slave next change ..... C-32
pending global transaction(s) ..... C-32
pipe get ..... C-32
pipe put ..... C-32
PL/SQL lock timer ..... C-33
pmon timer ..... C-33
prewarm transfer retry ..... C-33
prior process spawner to be cleaned up ..... C-33
process startup ..... C-34
PX Deque wait ..... C-34
PX qref latch ..... C-34
PX server shutdown ..... C-34
PX signal server ..... C-35
rdbms ipc message ..... C-35
rdbms ipc message block ..... C-35
rdbms ipc reply ..... C-35
read by other session ..... C-36
recovery read ..... C-36
Redo Transport Attach ..... C-36
Redo Transport Close ..... C-36

Redo Transport Detach ..... C-37
Redo Transport Open ..... C-37
Redo Transport Ping ..... C-37
Redo Transport Slave Shutdown ..... C-37
Redo Transport Slave Startup ..... C-37
Redo Writer Remote Sync Complete ..... C-37
Redo Write Remote Sync Notify ..... C-37
Remote Sync Ping ..... C-38
resmgr:become active ..... C-38
resmgr:cpu quantum ..... C-38
resmgr:pq queued ..... C-38
rolling migration: cluster quiesce ..... C-38
row cache lock ..... C-39
RVWR wait for flashback copy ..... C-39
sbtbufinfo ..... C-39
sbtgetbuf ..... C-39
sbtmapbuf ..... C-40
sbtrelbuf ..... C-40
scginq AST call ..... C-40
SGA: allocation forcing component growth ..... C-40
SGA: MMAN sleep for component shrink ..... C-40
SGA: sga_target resize ..... C-40
Shared IO Pool Memory ..... C-41
shared server idle wait ..... C-41
single-task message ..... C-41
smon timer ..... C-41
SQL*Net break/reset to client ..... C-41
SQL*Net break/reset to dblink ..... C-42
SQL*Net message from client ..... C-42
SQL*Net message from dblink ..... C-42
SQL*Net message to client ..... C-43
SQL*Net message to dblink ..... C-43
SQL*Net more data from client ..... C-43
SQL*Net more data from dblink ..... C-43
SQL*Net more data to client ..... C-44
SQL*Net more data to dblink ..... C-44
Streams AQ: waiting for messages in the queue ..... C-44
switch logfile command ..... C-44
SYNC Remote Write ..... C-45
TCP Socket (KGAS) ..... C-45
timer in sksawat ..... C-45
transaction ..... C-45
unbound tx ..... C-46

undo_retention publish retry ..... C-46
undo segment extension ..... C-46
undo segment recovery ..... C-46
undo segment tx slot ..... C-46
utl_file I/O ..... C-47
virtual circuit status ..... C-47
virtual circuit wait ..... C-47
WCR: replay client notify ..... C-47
WCR: replay clock ..... C-48
WCR: replay lock order ..... C-48
WCR: replay paused ..... C-48
WCR: Sync context busy ..... C-48
WMON goes to sleep ..... C-48
write complete waits ..... C-48
writes stopped by instance recovery or database suspension ..... C-49

D Oracle Enqueue Names

**E** Statistics Descriptions

Displaying Statistics ..... E-1
Statistics Descriptions ..... E-1

F Background Processes

**Index**

# Preface

This manual provides reference information about database initialization parameters, static data dictionary views, dynamic performance views, database limits, and SQL scripts that are part of the Oracle Database.

Oracle Database Reference contains information that describes the features and functionality of the Oracle Database (also known as the standard edition) and the Oracle Database Enterprise Edition products. The Oracle Database and the Oracle Database Enterprise Edition have the same basic features. However, several advanced features are available only with the Enterprise Edition, and some of these are optional. For example, to use application failover, you must have the Enterprise Edition with the Real Application Clusters option.

---

**See Also:**

*Oracle Database New Features Guide* for information about the differences between the Oracle Database and the Oracle Database Enterprise Edition and the features and options that are available to you.

---

This preface contains these topics:


- Audience
- Documentation Accessibility
- Related Documents
- Conventions


## Audience

Oracle Database Reference is intended for database administrators, system administrators, and database application developers.

To use this document, you need to be familiar with the following:

- Oracle database management system (DBMS) concepts
- Your operating system environment

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

## Related Documents

For more information, see these Oracle resources:

- *Oracle Database Concepts* for a comprehensive introduction to the concepts and terminology used in this manual
- *Oracle Database Administrator's Guide* for information about administering the Oracle Database
- *Oracle Database Upgrade Guide* for the procedures for upgrading a previous release of Oracle to the new release
- *Oracle Database SQL Language Reference* for information on Oracle's SQL commands and functions
- *Oracle Database Development Guide* for information about developing database applications within the Oracle Database

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

http://www.oracle.com/technetwork

You can go directly to the documentation section of the OTN Web site at

http://www.oracle.com/technetwork/documentation

## Conventions

The following text conventions are used in this document:

| Convention | Meaning |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

# Changes in This Release for Oracle Database Reference

This preface lists changes in *Oracle Database Reference*:


- Changes in Oracle Database 12c Release 1 (12.1.0.2)
- Changes in Oracle Database 12c Release 1 (12.1.0.1)


## Changes in Oracle Database 12*c* Release 1 (12.1.0.2)

The following are changes in *Oracle Database Reference* for Oracle Database 12*c* Release 1 (12.1.0.2).

### New Features

- The following initialization parameters are new in this release:

  ```
  COMMON_USER_PREFIX
  DB_BIG_TABLE_CACHE_PERCENT_TARGET
  DBFIPS_140
  ENABLE_GOLDENGATE_REPLICATION[1]
  INMEMORY_CLAUSE_DEFAULT
  INMEMORY_FORCE
  INMEMORY_MAX_POPULATE_SERVERS
  INMEMORY_QUERY
  INMEMORY_SIZE
  INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT
  OPTIMIZER_INMEMORY_AWARE
  ```

  **See Also:**

  Initialization Parameters for more information about these initialization parameters

- The following static data dictionary views are new in this release:

[1] This initialization parameter was introduced in Oracle Database 11g Release 2 (11.2.0.4), but is available for the first time in Oracle Database 12c with Oracle Database 12c Release 1 (12.1.0.2).

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| ALL_CLUSTERING_DIMENSIONS | DBA_CLUSTERING_DIMENSIONS | USER_CLUSTERING_DIMENSIONS |
| ALL_CLUSTERING_JOINS | DBA_CLUSTERING_JOINS | USER_CLUSTERING_JOINS |
| ALL_CLUSTERING_KEYS | DBA_CLUSTERING_KEYS | USER_CLUSTERING_KEYS |
| ALL_CLUSTERING_TABLES | DBA_CLUSTERING_TABLES | USER_CLUSTERING_TABLES |
| ALL_JSON_COLUMNS | DBA_JSON_COLUMNS | USER_JSON_COLUMNS |
| ALL_ZONEMAP_MEASURES | DBA_ZONEMAP_MEASURES | USER_ZONEMAP_MEASURES |
| ALL_ZONEMAPS | DBA_ZONEMAPS | USER_ZONEMAPS |
| | DBA_HIST_IM_SEG_STAT | |
| | DBA_HIST_IM_SEG_STAT_OBJ | |
| | DBA_PDB_SAVED_STATES | |
| | DBA_ROLLING_UNSUPPORTED | |

**See Also:**

Static Data Dictionary Views for more information about these static data dictionary views

- The following dynamic performance views are new in this release:

| GV$ Views | V$ Views |
|---|---|
| GV$BT_SCAN_CACHE | V$BT_SCAN_CACHE |
| GV$BT_SCAN_OBJ_TEMP | V$BT_SCAN_OBJ_TEMP |
| GV$EMX_USAGE_STATS | V$EMX_USAGE_STATS |
| GV$EVENT_HISTOGRAM_MICRO | V$EVENT_HISTOGRAM_MICRO |
| GV$FS_OBSERVER_HISTOGRAM | V$FS_OBSERVER_HISTOGRAM[1] |
| GV$IM_COLUMN_LEVEL | V$IM_COLUMN_LEVEL |
| GV$IM_SEGMENTS | V$IM_SEGMENTS |
| GV$IM_USER_SEGMENTS | V$IM_USER_SEGMENTS |
| GV$INMEMORY_AREA | V$INMEMORY_AREA |
| GV$KEY_VECTOR | V$KEY_VECTOR |

[1] This view was introduced in Oracle Database 11*g* Release 2 (11.2.0.4), but is available for the first time in Oracle Database 12*c* with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

Dynamic Performance Views for more information about these dynamic performance views

## Deprecated Features

The following features are deprecated in this release, and may be desupported in a future release:

- The RDBMS_SERVER_DN initialization parameter

  See "RDBMS_SERVER_DN" for information about alternatives.

# Changes in Oracle Database 12*c* Release 1 (12.1.0.1)

The following are changes in *Oracle Database Reference* for Oracle Database 12*c* Release 1 (12.1.0.1).

## New Features

- The following initialization parameters are new in this release:

  ```
  CONNECTION_BROKERS
  DNFS_BATCH_SIZE
  ENABLE_PLUGGABLE_DATABASE
  HEAT_MAP
  MAX_STRING_SIZE
  OPTIMIZER_ADAPTIVE_FEATURES
  OPTIMIZER_ADAPTIVE_REPORTING_ONLY
  PDB_FILE_NAME_CONVERT
  PGA_AGGREGATE_LIMIT
  PROCESSOR_GROUP_NAME
  NONCDB_COMPATIBLE
  SPATIAL_VECTOR_ACCELERATION
  TEMP_UNDO_ENABLED
  THREADED_EXECUTION
  UNIFIED_AUDIT_SGA_QUEUE_SIZE
  USE_DEDICATED_BROKER
  ```

  **See Also:**

  Initialization Parameters for more information about these initialization parameters

- The following static data dictionary views are new in this release:

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| ALL_APPLY_INSTANTIATED_GLOBAL | DBA_APPLY_INSTANTIATED_GLOBAL | |
| ALL_APPLY_INSTANTIATED_OBJECTS | DBA_APPLY_INSTANTIATED_OBJECTS | |
| ALL_APPLY_INSTANTIATED_SCHEMAS | DBA_APPLY_INSTANTIATED_SCHEMAS | |
| ALL_ATTRIBUTE_TRANSFORMATIONS | DBA_ATTRIBUTE_TRANSFORMATIONS | USER_ATTRIBUTE_TRANSFORMATIONS |
| ALL_CODE_ROLE_PRIVS | DBA_CODE_ROLE_PRIVS | USER_CODE_ROLE_PRIVS |
| ALL_CREDENTIALS | DBA_CREDENTIALS | USER_CREDENTIALS |
| ALL_CUBE_NAMED_BUILD_SPECS | DBA_CUBE_NAMED_BUILD_SPECS | USER_CUBE_NAMED_BUILD_SPECS |
| ALL_CUBE_SUB_PARTITION_LEVELS | DBA_CUBE_SUB_PARTITION_LEVELS | USER_CUBE_SUB_PARTITION_LEVELS |
| ALL_ERROR_TRANSLATIONS | DBA_ERROR_TRANSLATIONS | USER_ERROR_TRANSLATIONS |
| ALL_GG_INBOUND_PROGRESS | DBA_GG_INBOUND_PROGRESS | |
| ALL_GOLDENGATE_INBOUND | DBA_GOLDENGATE_INBOUND | |
| ALL_GOLDENGATE_PRIVILEGES | DBA_GOLDENGATE_PRIVILEGES | USER_GOLDENGATE_PRIVILEGES |
| ALL_GOLDENGATE_RULES | DBA_GOLDENGATE_RULES | |
| ALL_HEAT_MAP_SEG_HISTOGRAM | DBA_HEAT_MAP_SEG_HISTOGRAM | USER_HEAT_MAP_SEG_HISTOGRAM |
| ALL_HEAT_MAP_SEGMENT | DBA_HEAT_MAP_SEGMENT | USER_HEAT_MAP_SEGMENT |
| ALL_MEASURE_FOLDER_SUBFOLDERS | DBA_MEASURE_FOLDER_SUBFOLDERS | USER_MEASURE_FOLDER_SUBFOLDERS |
| ALL_METADATA_PROPERTIES | DBA_METADATA_PROPERTIES | USER_METADATA_PROPERTIES |
| ALL_PLSQL_COLL_TYPES | DBA_PLSQL_COLL_TYPES | USER_PLSQL_COLL_TYPES |
| ALL_PLSQL_TYPE_ATTRS | DBA_PLSQL_TYPE_ATTRS | USER_PLSQL_TYPE_ATTRS |
| ALL_PLSQL_TYPES | DBA_PLSQL_TYPES | USER_PLSQL_TYPES |
| ALL_POLICY_ATTRIBUTES | DBA_POLICY_ATTRIBUTES | USER_POLICY_ATTRIBUTES |

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| ALL_SQL_TRANSLATION_PROFILES | DBA_SQL_TRANSLATION_PROFILES | USER_SQL_TRANSLATION_PROFILES |
| ALL_SQL_TRANSLATIONS | DBA_SQL_TRANSLATIONS | USER_SQL_TRANSLATIONS |
| ALL_TAB_IDENTITY_COLS | DBA_TAB_IDENTITY_COLS | USER_TAB_IDENTITY_COLS |
| ALL_TRANSFORMATIONS | DBA_TRANSFORMATIONS | USER_TRANSFORMATIONS |
| ALL_XML_NESTED_TABLES | DBA_XML_NESTED_TABLES | USER_XML_NESTED_TABLES |
| ALL_XML_OUT_OF_LINE_TABLES | DBA_XML_OUT_OF_LINE_TABLES | USER_XML_OUT_OF_LINE_TABLES |
| ALL_XML_SCHEMA_ATTRIBUTES | DBA_XML_SCHEMA_ATTRIBUTES | USER_XML_SCHEMA_ATTRIBUTES |
| ALL_XML_SCHEMA_COMPLEX_TYPES | DBA_XML_SCHEMA_COMPLEX_TYPES | USER_XML_SCHEMA_COMPLEX_TYPES |
| ALL_XML_SCHEMA_ELEMENTS | DBA_XML_SCHEMA_ELEMENTS | USER_XML_SCHEMA_ELEMENTS |
| ALL_XML_SCHEMA_NAMESPACES | DBA_XML_SCHEMA_NAMESPACES | USER_XML_SCHEMA_NAMESPACES |
| ALL_XML_SCHEMA_SIMPLE_TYPES | DBA_XML_SCHEMA_SIMPLE_TYPES | USER_XML_SCHEMA_SIMPLE_TYPES |
| ALL_XML_SCHEMA_SUBSTGRP_HEAD | DBA_XML_SCHEMA_SUBSTGRP_HEAD | USER_XML_SCHEMA_SUBSTGRP_HEAD |
| ALL_XML_SCHEMA_SUBSTGRP_MBRS | DBA_XML_SCHEMA_SUBSTGRP_MBRS | USER_XML_SCHEMA_SUBSTGRP_MBRS |
| ALL_XSTREAM_ADMINISTRATOR | DBA_XSTREAM_ADMINISTRATOR | |
| ALL_XSTREAM_OUT_SUPPORT_MODE | DBA_XSTREAM_OUT_SUPPORT_MODE | |
| ALL_XSTREAM_TRANSFORMATIONS | DBA_XSTREAM_TRANSFORMATIONS | |
| AUDIT_UNIFIED_CONTEXTS | | |
| AUDIT_UNIFIED_ENABLED_POLICIES | | |
| AUDIT_UNIFIED_POLICIES | | |

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| AUDIT_UNIFIED_POLICY_COMMENTS | | |
| AUDITABLE_SYSTEM_ACTIONS | | |
| DATABASE_EXPORT_OBJECTS | | |
| | DBA_ACL_NAME_MAP | |
| | DBA_ALERT_HISTORY_DETAIL | |
| | DBA_CDB_RSRC_PLAN_DIRECTIVES | |
| | DBA_CDB_RSRC_PLANS | |
| | DBA_CONTAINER_DATA | |
| | DBA_DIGEST_VERIFIERS | |
| | DBA_DISCOVERY_SOURCE | |
| | DBA_EDITIONED_TYPES | USER_EDITIONED_TYPES |
| | DBA_GOLDENGATE_SUPPORT_MODE | |
| | DBA_HEATMAP_TOP_OBJECTS | |
| | DBA_HEATMAP_TOP_TABLESPACES | |
| | DBA_HIST_APPLY_SUMMARY | |
| | DBA_HIST_ASH_SNAPSHOT | |
| | DBA_HIST_CAPTURE | |
| | DBA_HIST_PDB_INSTANCE | |
| | DBA_HIST_REPLICATION_TBL_STATS | |
| | DBA_HIST_REPLICATION_TXN_STATS | |
| | DBA_HIST_REPORTS | |
| | DBA_HIST_REPORTS_DETAILS | |

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| | DBA_HIST_REPORTS_TIMEBANDS | |
| | DBA_HIST_SESS_SGA_STATS | |
| | DBA_HIST_TABLESPACE | |
| | DBA_HOST_ACES | USER_HOST_ACES |
| | DBA_HOST_ACLS | |
| | DBA_ILMDATAMOVEMENTPOLICIES | USER_ILMDATAMOVEMENTPOLICIES |
| | DBA_ILMEVALUATIONDETAILS | USER_ILMEVALUATIONDETAILS |
| | DBA_ILMOBJECTS | USER_ILMOBJECTS |
| | DBA_ILMPARAMETERS | |
| | DBA_ILMPOLICIES | USER_ILMPOLICIES |
| | DBA_ILMRESULTS | USER_ILMRESULTS |
| | DBA_ILMTASKS | USER_ILMTASKS |
| | DBA_LOGSTDBY_EDS_SUPPORTED | |
| | DBA_LOGSTDBY_EDS_TABLES | |
| | DBA_LOGSTDBY_PLSQL_MAP | |
| | DBA_LOGSTDBY_PLSQL_SUPPORT | |
| | DBA_MINING_MODEL_TABLES | |
| | DBA_OBJECT_USAGE | USER_OBJECT_USAGE |
| | DBA_OPTSTAT_OPERATION_TASKS | |
| | DBA_PDB_HISTORY | |
| | DBA_PDBS | |
| | DBA_PRIV_CAPTURES | |
| | DBA_ROLLING_DATABASES | |

| ALL_ Views | DBA_ Views | USER_ Views |
| --- | --- | --- |
| | DBA_ROLLING_EVENTS | |
| | DBA_ROLLING_PARAMETERS | |
| | DBA_ROLLING_PLAN | |
| | DBA_ROLLING_STATISTICS | |
| | DBA_ROLLING_STATUS | |
| | DBA_SENSITIVE_COLUMN_TYPES | |
| | DBA_SENSITIVE_COLUMNS | |
| | DBA_SQL_PLAN_DIR_OBJECTS | |
| | DBA_SQL_PLAN_DIRECTIVES | |
| | DBA_SR_GRP_STATUS | USER_SR_GRP_STATUS |
| | DBA_SR_GRP_STATUS_ALL | USER_SR_GRP_STATUS_ALL |
| | DBA_SR_OBJ | USER_SR_OBJ |
| | DBA_SR_OBJ_ALL | USER_SR_OBJ_ALL |
| | DBA_SR_OBJ_STATUS | USER_SR_OBJ_STATUS |
| | DBA_SR_OBJ_STATUS_ALL | USER_SR_OBJ_STATUS_ALL |
| | DBA_SR_PARTN_OPS | USER_SR_PARTN_OPS |
| | DBA_SR_STLOG_EXCEPTIONS | USER_SR_STLOG_EXCEPTIONS |
| | DBA_SR_STLOG_STATS | USER_SR_STLOG_STATS |
| | DBA_SUPPLEMENTAL_LOGGING | |
| | DBA_TSDP_IMPORT_ERRORS | |
| | DBA_TSDP_POLICY_CONDITION | |
| | DBA_TSDP_POLICY_FEATURE | |

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| | DBA_TSDP_POLICY_PARAMETER | |
| | DBA_TSDP_POLICY_PROTECTION | |
| | DBA_TSDP_POLICY_TYPE | |
| | DBA_UNUSED_OBJPRIVS | |
| | DBA_UNUSED_OBJPRIVS_PATH | |
| | DBA_UNUSED_PRIVS | |
| | DBA_UNUSED_SYSPRIVS | |
| | DBA_UNUSED_SYSPRIVS_PATH | |
| | DBA_UNUSED_USERPRIVS | |
| | DBA_UNUSED_USERPRIVS_PATH | |
| | DBA_USED_OBJPRIVS | |
| | DBA_USED_OBJPRIVS_PATH | |
| | DBA_USED_PRIVS | |
| | DBA_USED_PUBPRIVS | |
| | DBA_USED_SYSPRIVS | |
| | DBA_USED_SYSPRIVS_PATH | |
| | DBA_USED_USERPRIVS | |
| | DBA_USED_USERPRIVS_PATH | |
| | DBA_WALLET_ACES | USER_WALLET_ACES |
| | DBA_WI_CAPTURE_FILES | |
| | DBA_WI_JOBS | |
| | DBA_WI_OBJECTS | |
| | DBA_WI_PATTERN_ITEMS | |
| | DBA_WI_PATTERNS | |

| ALL_ Views | DBA_ Views | USER_ Views |
| --- | --- | --- |
| | DBA_WI_STATEMENTS | |
| | DBA_WI_TEMPLATE_EXECUTIONS | |
| | DBA_WI_TEMPLATES | |
| | DBA_WORKLOAD_ACTIVE_USER_MAP | |
| | DBA_WORKLOAD_REPLAY_SCHEDULES | |
| | DBA_WORKLOAD_SCHEDULE_CAPTURES | |
| | DBA_WORKLOAD_SCHEDULE_ORDERING | |
| | DBA_WORKLOAD_USER_MAP | |
| | DBA_XS_AUDIT_POLICY_OPTIONS | |
| | DBA_XS_AUDIT_TRAIL | |
| | DBA_XS_ENB_AUDIT_POLICIES | |
| | DBA_XSTREAM_SPLIT_MERGE | |
| | DBA_XSTREAM_SPLIT_MERGE_HIST | |
| | DBA_XSTREAM_STMT_HANDLERS | |
| | DBA_XSTREAM_STMTS | |
| PDB_PLUG_IN_VIOLATIONS | | |
| REDACTION_COLUMNS | | |
| REDACTION_POLICIES | | |
| REDACTION_VALUES_FOR_TYPE_FULL | | |
| SCHEMA_EXPORT_OBJECTS | | |
| TABLE_EXPORT_OBJECTS | | |
| UNIFIED_AUDIT_TRAIL | | |

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| | | USER_PRIVILEGE_MAP |

**See Also:**

Static Data Dictionary Views for more information about these static data dictionary views

- There are new CDB views introduced in this release.

**See Also:**

"CDB_* Views" for more information about CDB views

- The following dynamic performance views are new in this release:

| GV$ Views | V$ Views |
|---|---|
| GV$AQ_BACKGROUND_COORDINATOR | V$AQ_BACKGROUND_COORDINATOR |
| GV$AQ_BMAP_NONDUR_SUBSCRIBERS | V$AQ_BMAP_NONDUR_SUBSCRIBERS |
| GV$AQ_CROSS_INSTANCE_JOBS | V$AQ_CROSS_INSTANCE_JOBS |
| GV$AQ_JOB_COORDINATOR | V$AQ_JOB_COORDINATOR |
| GV$AQ_NONDUR_REGISTRATIONS | V$AQ_NONDUR_REGISTRATIONS |
| GV$AQ_NONDUR_SUBSCRIBER | V$AQ_NONDUR_SUBSCRIBER |
| GV$AQ_NONDUR_SUBSCRIBER_LWM | V$AQ_NONDUR_SUBSCRIBER_LWM |
| GV$AQ_NOTIFICATION_CLIENTS | V$AQ_NOTIFICATION_CLIENTS |
| GV$AQ_SERVER_POOL | V$AQ_SERVER_POOL |
| GV$AQ_SUBSCRIBER_LOAD | V$AQ_SUBSCRIBER_LOAD |
| GV$ASM_ACFS_SEC_ADMIN | V$ASM_ACFS_SEC_ADMIN |
| GV$ASM_ACFS_SEC_CMDRULE | V$ASM_ACFS_SEC_CMDRULE |
| GV$ASM_ACFS_SEC_REALM | V$ASM_ACFS_SEC_REALM |
| GV$ASM_ACFS_SEC_REALM_FILTER | V$ASM_ACFS_SEC_REALM_FILTER |
| GV$ASM_ACFS_SEC_REALM_GROUP | V$ASM_ACFS_SEC_REALM_GROUP |
| GV$ASM_ACFS_SEC_REALM_USER | V$ASM_ACFS_SEC_REALM_USER |
| GV$ASM_ACFS_SEC_RULE | V$ASM_ACFS_SEC_RULE |
| GV$ASM_ACFS_SEC_RULESET | V$ASM_ACFS_SEC_RULESET |
| GV$ASM_ACFS_SEC_RULESET_RULE | V$ASM_ACFS_SEC_RULESET_RULE |

| GV$ Views | V$ Views |
|---|---|
| GV$ASM_ACFSREPL | V$ASM_ACFSREPL |
| GV$ASM_ACFSREPLTAG | V$ASM_ACFSREPLTAG |
| GV$ASM_ACFSTAG | V$ASM_ACFSTAG |
| GV$ASM_ESTIMATE | V$ASM_ESTIMATE |
| GV$BACKUP_NONLOGGED | V$BACKUP_NONLOGGED |
| GV$CLIENT_SECRETS | V$CLIENT_SECRETS |
| GV$CLONEDFILE | V$CLONEDFILE |
| GV$CON_SYS_TIME_MODEL | V$CON_SYS_TIME_MODEL |
| GV$CON_SYSSTAT | V$CON_SYSSTAT |
| GV$CON_SYSTEM_EVENT | V$CON_SYSTEM_EVENT |
| GV$CON_SYSTEM_WAIT_CLASS | V$CON_SYSTEM_WAIT_CLASS |
| GV$CONTAINERS | V$CONTAINERS |
| GV$COPY_NONLOGGED | V$COPY_NONLOGGED |
| GV$DEAD_CLEANUP | V$DEAD_CLEANUP |
| GV$DG_BROKER_CONFIG | V$DG_BROKER_CONFIG |
| GV$EDITIONABLE_TYPES | V$EDITIONABLE_TYPES |
| GV$ENCRYPTION_KEYS | V$ENCRYPTION_KEYS |
| GV$FLASHFILESTAT | V$FLASHFILESTAT |
| GV$GG_APPLY_COORDINATOR | V$GG_APPLY_COORDINATOR |
| GV$GG_APPLY_READER | V$GG_APPLY_READER |
| GV$GG_APPLY_SERVER | V$GG_APPLY_SERVER |
| GV$GOLDENGATE_CAPTURE | V$GOLDENGATE_CAPTURE |
| GV$GOLDENGATE_MESSAGE_TRACKING | V$GOLDENGATE_MESSAGE_TRACKING |
| GV$GOLDENGATE_TABLE_STATS | V$GOLDENGATE_TABLE_STATS |
| GV$GOLDENGATE_TRANSACTION | V$GOLDENGATE_TRANSACTION |
| GV$HEAT_MAP_SEGMENT | V$HEAT_MAP_SEGMENT |
| GV$INSTANCE_PING | V$INSTANCE_PING |
| GV$IO_OUTLIER | V$IO_OUTLIER |
| GV$LGWRIO_OUTLIER | V$LGWRIO_OUTLIER |

| GV$ Views | V$ Views |
|---|---|
| GV$MAPPED_SQL | V$MAPPED_SQL |
| GV$NONLOGGED_BLOCK | V$NONLOGGED_BLOCK |
| GV$OPTIMIZER_PROCESSING_RATE | V$OPTIMIZER_PROCESSING_RATE |
| GV$PATCHES | V$PATCHES |
| GV$PDB_INCARNATION | V$PDB_INCARNATION |
| GV$PDBS | V$PDBS |
| GV$SESSIONS_COUNT | V$SESSIONS_COUNT |
| GV$RO_USER_ACCOUNT | V$RO_USER_ACCOUNT |
| GV$SQL_MONITOR_SESSTAT | V$SQL_MONITOR_SESSTAT |
| GV$SQL_MONITOR_STATNAME | V$SQL_MONITOR_STATNAME |
| GV$TEMPUNDOSTAT | V$TEMPUNDOSTAT |
| GV$XSTREAM_APPLY_COORDINATOR | V$XSTREAM_APPLY_COORDINATOR |
| GV$XSTREAM_APPLY_READER | V$XSTREAM_APPLY_READER |
| GV$XSTREAM_APPLY_SERVER | V$XSTREAM_APPLY_SERVER |
| GV$XSTREAM_TABLE_STATS | V$XSTREAM_TABLE_STATS |

**See Also:**

Dynamic Performance Views for more information about these dynamic performance views

## Deprecated Features

The following features are deprecated in this release, and may be desupported in a future release:

- The `FILE_MAPPING` initialization parameter has been deprecated.

  See "FILE_MAPPING" for more information.

- The `SEC_CASE_SENSITIVE_LOGON` initialization parameter has been deprecated.

  See *Oracle Database Security Guide* for more information.

- The `ALL_SCHEDULER_CREDENTIALS` view has been deprecated.

  See "ALL_SCHEDULER_CREDENTIALS" for information about alternatives.

- The `DBA_NETWORK_ACL_PRIVILEGES` view has been deprecated.

  See "DBA_NETWORK_ACL_PRIVILEGES" for information about alternatives.

- The `DBA_NETWORK_ACLS` view has been deprecated.

See “DBA_NETWORK_ACLS” for information about alternatives.

- The `DBA_SCHEDULER_CREDENTIALS` view has been deprecated.
  See “DBA_SCHEDULER_CREDENTIALS” for information about alternatives.
- The `USER_NETWORK_ACL_PRIVILEGES` view has been deprecated.
  See “USER_NETWORK_ACL_PRIVILEGES” for information about alternatives.
- The `USER_SCHEDULER_CREDENTIALS` view has been deprecated.
  See “USER_SCHEDULER_CREDENTIALS” for information about alternatives.
- The `V$OBJECT_USAGE` view has been deprecated.
  See “V$OBJECT_USAGE” for information about alternatives.

# Part I

## Initialization Parameters

This part describes the database initialization parameters that can be specified in a parameter file to start or configure an instance. This part contains the following chapter:

- Initialization Parameters

# 1

# Initialization Parameters

This chapter contains detailed descriptions (in alphabetical order) of the database initialization parameters.

This chapter contains the following topics:


- Uses of Initialization Parameters
- Basic Initialization Parameters
- Parameter Files
- Changing Parameter Values in a Parameter File
- Reading the Parameter Descriptions
- Initialization Parameter Descriptions


## Uses of Initialization Parameters

Initialization parameters fall into various functional groups. For example, parameters perform the following functions:

- Set limits for the entire database
- Set user or process limits
- Set limits on database resources
- Affect performance (these are called **variable parameters**)

  Variable parameters are of particular interest to database administrators, because these parameters are used primarily to improve database performance.

Database administrators can use initialization parameters to:

- Optimize performance by adjusting memory structures, such as the number of database buffers in memory
- Set database-wide defaults, such as the amount of space initially allocated for a context area when it is created
- Set database limits, such as the maximum number of database users
- Specify names of files or directories required by the database

Many initialization parameters can be fine-tuned to improve database performance. Other parameters should never be altered or should be altered only under the supervision of Oracle Support Services.

All initialization parameters are optional. Oracle has a default value for each parameter. This value may be operating system-dependent, depending on the parameter.

## Types of Initialization Parameters

The Oracle database server has the following types of initialization parameters:

- Derived Parameters
- Operating System-Dependent Parameters
- Variable Parameters (these can be dynamic parameters or any of the preceding ones)

### Derived Parameters

Some initialization parameters are **derived**, meaning that their values are calculated from the values of other parameters. Normally, you should not alter values for derived parameters, but if you do, then the value you specify will override the calculated value.

For example, the default value of the `SESSIONS` parameter is derived from the value of the `PROCESSES` parameter. If the value of `PROCESSES` changes, then the default value of `SESSIONS` changes as well, unless you override it with a specified value.

### Operating System-Dependent Parameters

The valid values or value ranges of some initialization parameters depend upon the host operating system. For example, the parameter `DB_BLOCK_BUFFERS` indicates the number of data buffers in main memory, and its maximum value depends on the operating system. The size of those buffers, set by `DB_BLOCK_SIZE`, has an operating system-dependent default value.

---

**See Also:**

Your operating system-specific Oracle documentation for more information on operating system-dependent Oracle parameters and operating system parameters

---

### Variable Parameters

The variable initialization parameters offer the most potential for improving system performance. Some variable parameters set capacity limits but do not affect performance. For example, when the value of `OPEN_CURSORS` is 10, a user process attempting to open its eleventh cursor receives an error. Other variable parameters affect performance but do not impose absolute limits. For example, reducing the value of `DB_BLOCK_BUFFERS` does not prevent work even though it may slow down performance.

Increasing the values of variable parameters may improve your system's performance, but increasing most parameters also increases the system global area (SGA) size. A larger SGA can improve database performance up to a point. In virtual memory operating systems, an SGA that is too large can degrade performance if it is swapped in and out of memory. Operating system parameters that control virtual memory working areas should be set with the SGA size in mind. The operating system configuration can also limit the maximum size of the SGA.

## Basic Initialization Parameters

The following is a list of the database basic initialization parameters. Most databases should only need to have basic parameters set to run properly and efficiently. Oracle advises you to become familiar with the basic parameters and only use other parameters when directed to by feature documentation or in special circumstances:

CLUSTER_DATABASE
COMPATIBLE
CONTROL_FILES
DB_BLOCK_SIZE
DB_CREATE_FILE_DEST
DB_CREATE_ONLINE_LOG_DEST_*n*
DB_DOMAIN
DB_NAME
DB_RECOVERY_FILE_DEST
DB_RECOVERY_FILE_DEST_SIZE
DB_UNIQUE_NAME
INSTANCE_NUMBER
LDAP_DIRECTORY_SYSAUTH
LOG_ARCHIVE_DEST_*n*
LOG_ARCHIVE_DEST_STATE_*n*
NLS_LANGUAGE
NLS_TERRITORY
OPEN_CURSORS
PGA_AGGREGATE_TARGET
PROCESSES
REMOTE_LISTENER
REMOTE_LOGIN_PASSWORDFILE
SESSIONS
SGA_TARGET
SHARED_SERVERS
STAR_TRANSFORMATION_ENABLED
UNDO_TABLESPACE

## Parameter Files

A **parameter file** is a file that contains a list of initialization parameters and a value for each parameter. You specify initialization parameters in a parameter file that reflect your particular installation. Oracle supports the following two types of parameter files:

- Server Parameter Files
- Initialization Parameter Files

### Server Parameter Files

A **server parameter file** is a binary file that acts as a repository for initialization parameters. The server parameter file can reside on the computer where the Oracle database server executes. Initialization parameters stored in a server parameter file are

persistent, in that any changes made to the parameters while an instance is running can persist across instance shutdown and startup.

**See Also:**

*Oracle Database Administrator's Guide* for an introduction to and detailed information about managing and using server parameter files and initialization parameter files

## Initialization Parameter Files

An **initialization parameter file** is a text file that contains a list of initialization parameters. The file should be written in the client's default character set.

The following are sample entries in an initialization parameter file:

```
PROCESSES = 100
OPEN_LINKS = 12
GLOBAL_NAMES = true
```

The name of the initialization parameter file varies depending on the operating system. For example, it can be in mixed case or lowercase, or it can have a logical name or a variation of the name `init.ora`. Also supplied is an `initdw.ora` file, which contains suggested parameter settings for data warehouses and data marts. The database administrator can choose a different file name for the initialization parameter file.

Refer to your operating system-specific Oracle documentation for the default locations and filenames for initialization parameter files on your operating system. The initialization parameter file is read by the client-side tool used to start the server (such as SQL*Plus).

Sample initialization parameter files are provided on the Oracle distribution medium for each operating system. A sample file is sufficient for initial use, but you will probably want to modify the file to tune the database for best performance. Any changes will take effect after you completely shut down and restart the instance.

### Specifying Values in an Initialization Parameter File

This section describes several aspects of setting parameter values in an initialization parameter file.

#### Rules Governing Initialization Parameter Files

The following rules govern the specification of parameters in initialization parameter files:

- An initialization parameter file should contain only parameters and comments. A pound sign (#) starts a comment line. The rest of the line is ignored.
- You can specify parameters in any order.
- Case (upper or lower) in filenames is significant only if case is significant on the host operating system.
- To enter several parameters on one line, use spaces between parameter names and values, as in the following example:

  ```
  PROCESSES = 100 CPU_COUNT = 1 OPEN_CURSORS = 10
  ```

- Some parameters, such as ROLLBACK_SEGMENTS, accept multiple values. Any of the following represent valid syntax.

  – Enter multiple values enclosed in parentheses and separated by commas. For example:

    ```
    ROLLBACK_SEGMENTS = (SEG1, SEG2, SEG3, SEG4, SEG5)
    ```

  – Enter multiple values without parentheses and commas. For example:

    ```
    ROLLBACK_SEGMENTS = SEG1 SEG2 SEG3 SEG4 SEG5
    ```

  – Enter multiple values, one per line. For example:

    ```
    ROLLBACK_SEGMENTS = SEG1
    ROLLBACK_SEGMENTS = SEG2
    ROLLBACK_SEGMENTS = SEG3
    ROLLBACK_SEGMENTS = SEG4
    ROLLBACK_SEGMENTS = SEG5
    ```

  If you enter values for one parameter on multiple lines, then the entries must be on consecutive lines. If they are not, then the first entry will not be processed properly. For example, in the following entry the setting for SEG3 and SEG4 will override the setting for SEG1 and SEG2:

  ```
  ROLLBACK_SEGMENTS = SEG1 SEG2
  OPEN_CURSORS = 10
  ROLLBACK_SEGMENTS = SEG3 SEG4
  ```

- A backslash (\), also known as an escape character, indicates continuation of the parameter specification. If a backslash continues a line, then the continued line must have no leading spaces. For example:

  ```
  ROLLBACK_SEGMENTS = (SEG1, SEG2, \
  SEG3, SEG4, SEG5)
  ```

- You can use the IFILE initialization parameter to embed the contents of another initialization parameter file into the current initialization parameter file.

- Enclose in quotation marks any parameter values that contain spaces or tabs. You can use either single or double quotation marks unless otherwise indicated. For example:

  ```
  NLS_TERRITORY = 'CZECH REPUBLIC'
  ```

  **Note:**

  Listing parameters in alphabetical order in the initialization parameter file can help you to find them and can help ensure that each parameter is specified only once.

- Enclose in quotation marks any parameter value that contains a special character.

  **See Also:**

  - Your operating system-specific Oracle documentation for more information on initialization parameter files
  - "IFILE"

### Using Special Characters in Parameter Values

If a parameter value contains a special character, then the special character must be preceded by a backslash or the entire parameter value must be enclosed in quotation marks. For example, you can specify special characters using either of the following:

```
DB_DOMAIN = 'JAPAN.ACME#.COM'

DB_DOMAIN = JAPAN.ACME\#.COM
```

Table 1-1 lists the special characters you can use in initialization parameter files.

***Table 1-1 Special Characters in Initialization Parameter Files***

| Character | Name | Description |
|---|---|---|
| # | Number sign | Comment |
| ( | Left parenthesis | Start list of values |
| ) | Right parenthesis | End list of values |
| " | Double quotation mark | Start or end of quoted string |
| ' | Single quotation mark | Start or end of quoted string |
| = | Equal sign | Separator of keyword and values |
| , | Comma | Separator of elements |
| - | Minus sign | Precedes UNIX-style keywords |
| \ | Backslash | Escape character |

If a special character must be treated literally in an initialization parameter file, then it must either be preceded by the backslash character, or the entire string containing the special character must be enclosed in quotation marks.

### Using the Escape Character

As described in "Rules Governing Initialization Parameter Files", the backslash (\) can also signify a line continuation. If the backslash is followed by an alphanumeric character, then the backslash is treated as a normal character in the input. If it is not followed by an alphanumeric character, then the backslash is treated either as a backslash or as a continuation character.

### Using Quotation Marks

Quotation marks can be nested in one of two ways. The first method is to double the quotation marks in the nested string. For example:

```
NLS_DATE_FORMAT = '''Today is'' MM/DD/YYYY'
```

The second method is to alternate single and double quotation marks. For example:

```
NLS_DATE_FORMAT = '"Today is" MM/DD/YYYY'
```

# Changing Parameter Values in a Parameter File

You change the value of a parameter in a parameter file in one of the following ways:

- By editing an initialization parameter file

  In most cases, the new value takes effect the next time you start an instance of the database.

- By issuing an `ALTER SYSTEM SET ... SCOPE=SPFILE` statement to update a server parameter file
- By issuing an `ALTER SYSTEM RESET ... SCOPE=SPFILE` statement to remove a parameter from a server parameter file, causing the default value to take effect the next time you start an instance of the database.

## Parameters by Functional Category

The following list shows the initialization parameters by their functional category:

- **ANSI Compliance**

  `BLANK_TRIMMING`

- **Backup and Restore**

  `BACKUP_TAPE_IO_SLAVES`
  `CLONEDB`
  `RECYCLEBIN`
  `TAPE_ASYNCH_IO`

- **BFILEs**

  `SESSION_MAX_OPEN_FILES`

- **Buffer Cache and I/O**

  `CLIENT_RESULT_CACHE_LAG`
  `CLIENT_RESULT_CACHE_SIZE`
  `DB_`*`n`*`K_CACHE_SIZE`
  `DB_BIG_TABLE_CACHE_PERCENT_TARGET`
  `DB_BLOCK_BUFFERS`
  `DB_BLOCK_SIZE`
  `DB_CACHE_ADVICE`
  `DB_CACHE_SIZE`
  `DB_FILE_MULTIBLOCK_READ_COUNT`
  `DB_KEEP_CACHE_SIZE`
  `DB_RECYCLE_CACHE_SIZE`
  `DB_WRITER_PROCESSES`
  `DBWR_IO_SLAVES`
  `DISK_ASYNCH_IO`
  `DNFS_BATCH_SIZE`

```
FILESYSTEMIO_OPTIONS
READ_ONLY_OPEN_DELAYED
RESULT_CACHE_MAX_RESULT
RESULT_CACHE_MAX_SIZE
RESULT_CACHE_MODE
USE_INDIRECT_DATA_BUFFERS
```

- **Cursors and Library Cache**

```
CURSOR_BIND_CAPTURE_DESTINATION
CURSOR_SHARING
CURSOR_SPACE_FOR_TIME
OPEN_CURSORS
SESSION_CACHED_CURSORS
```

- **Database/Instance Identification**

```
DB_DOMAIN
DB_NAME
INSTANCE_NAME
```

- **Diagnostics and Statistics**

```
BACKGROUND_CORE_DUMP
BACKGROUND_DUMP_DEST
CORE_DUMP_DEST
DB_BLOCK_CHECKING
DB_BLOCK_CHECKSUM
DIAGNOSTIC_DEST
EVENT
HEAT_MAP
MAX_DUMP_FILE_SIZE
SHADOW_CORE_DUMP
STATISTICS_LEVEL
TIMED_OS_STATISTICS
TIMED_STATISTICS
TRACE_ENABLED
TRACEFILE_IDENTIFIER
USER_DUMP_DEST
```

- **Distributed, Replication**

```
COMMIT_POINT_STRENGTH
DISTRIBUTED_LOCK_TIMEOUT
ENABLE_GOLDENGATE_REPLICATION
GLOBAL_NAMES
HS_AUTOREGISTER
OPEN_LINKS
OPEN_LINKS_PER_INSTANCE
```

REPLICATION_DEPENDENCY_TRACKING

- **File Locations, Names, and Sizes**

  AUDIT_FILE_DEST
  BACKGROUND_CORE_DUMP
  BACKGROUND_DUMP_DEST
  CONTROL_FILES
  CORE_DUMP_DEST
  DB_CREATE_FILE_DEST
  DB_CREATE_ONLINE_LOG_DEST_*n*
  DB_FILES
  DB_RECOVERY_FILE_DEST
  DB_RECOVERY_FILE_DEST_SIZE
  FILE_MAPPING
  IFILE
  LOG_ARCHIVE_DEST_*n*
  SPFILE

- **Globalization**

  NLS_CALENDAR
  NLS_COMP
  NLS_CURRENCY
  NLS_DATE_FORMAT
  NLS_DATE_LANGUAGE
  NLS_DUAL_CURRENCY
  NLS_ISO_CURRENCY
  NLS_LANGUAGE
  NLS_LENGTH_SEMANTICS
  NLS_NCHAR_CONV_EXCP
  NLS_NUMERIC_CHARACTERS
  NLS_SORT
  NLS_TERRITORY
  NLS_TIMESTAMP_FORMAT
  NLS_TIMESTAMP_TZ_FORMAT

- **Java**

  JAVA_JIT_ENABLED
  JAVA_MAX_SESSIONSPACE_SIZE
  JAVA_POOL_SIZE
  JAVA_SOFT_SESSIONSPACE_LIMIT

- **Job Queues**

  JOB_QUEUE_PROCESSES

- **License Limits**

```
LICENSE_MAX_SESSIONS
LICENSE_MAX_USERS
LICENSE_SESSIONS_WARNING
```

- **Memory**

```
INMEMORY_CLAUSE_DEFAULT
INMEMORY_FORCE
INMEMORY_MAX_POPULATE_SERVERS
INMEMORY_QUERY
INMEMORY_SIZE
INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT
MEMORY_MAX_TARGET
MEMORY_TARGET
```

- **Miscellaneous**

```
AQ_TM_PROCESSES
AWR_SNAPSHOT_TIME_OFFSET
ASM_PREFERRED_READ_FAILURE_GROUPS
COMPATIBLE
DB_INDEX_COMPRESSION_INHERITANCE
FIXED_DATE
LDAP_DIRECTORY_SYSAUTH
MAX_STRING_SIZE
SPATIAL_VECTOR_ACCELERATION
XML_DB_EVENTS
```

- **Multitenant Architecture**

```
COMMON_USER_PREFIX
NONCDB_COMPATIBLE
PDB_FILE_NAME_CONVERT
```

- **Networking**

```
CONNECTION_BROKERS
LOCAL_LISTENER
REMOTE_LISTENER
SERVICE_NAMES
```

- **Objects and LOBs**

```
OBJECT_CACHE_MAX_SIZE_PERCENT
OBJECT_CACHE_OPTIMAL_SIZE
```

- **OLAP**

```
OLAP_PAGE_POOL_SIZE
```

- **Optimizer**

  OPTIMIZER_ADAPTIVE_FEATURES
  OPTIMIZER_ADAPTIVE_REPORTING_ONLY
  OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES
  OPTIMIZER_DYNAMIC_SAMPLING
  OPTIMIZER_FEATURES_ENABLE
  OPTIMIZER_INDEX_CACHING
  OPTIMIZER_INDEX_COST_ADJ
  OPTIMIZER_INMEMORY_AWARE
  OPTIMIZER_MODE
  OPTIMIZER_SECURE_VIEW_MERGING
  OPTIMIZER_USE_PENDING_STATISTICS
  OPTIMIZER_USE_SQL_PLAN_BASELINES
  QUERY_REWRITE_ENABLED
  QUERY_REWRITE_INTEGRITY
  STAR_TRANSFORMATION_ENABLED

- **Parallel Execution**

  PARALLEL_ADAPTIVE_MULTI_USER
  PARALLEL_EXECUTION_MESSAGE_SIZE
  PARALLEL_MAX_SERVERS
  PARALLEL_MIN_PERCENT
  PARALLEL_MIN_SERVERS
  PARALLEL_THREADS_PER_CPU

- **PL/SQL**

  PLSQL_V2_COMPATIBILITY
  REMOTE_DEPENDENCIES_MODE
  UTL_FILE_DIR

- **PL/SQL Compiler**

  PERMIT_92_WRAP_FORMAT
  PLSCOPE_SETTINGS
  PLSQL_CCFLAGS
  PLSQL_CODE_TYPE
  PLSQL_DEBUG
  PLSQL_OPTIMIZE_LEVEL
  PLSQL_WARNINGS
  NLS_LENGTH_SEMANTICS

- **SGA Memory**

  DB_*n*K_CACHE_SIZE
  DB_CACHE_SIZE
  HI_SHARED_MEMORY_ADDRESS
  JAVA_POOL_SIZE

LARGE_POOL_SIZE
LOCK_SGA
OLAP_PAGE_POOL_SIZE
PRE_PAGE_SGA
SGA_MAX_SIZE
SGA_TARGET
SHARED_MEMORY_ADDRESS
SHARED_POOL_RESERVED_SIZE
SHARED_POOL_SIZE
STREAMS_POOL_SIZE
USE_LARGE_PAGES

- **Oracle RAC**

  ACTIVE_INSTANCE_COUNT
  CLUSTER_DATABASE
  CLUSTER_DATABASE_INSTANCES
  CLUSTER_INTERCONNECTS
  INSTANCE_NUMBER
  PARALLEL_INSTANCE_GROUP
  THREAD

- **Redo Logs, Archiving, and Recovery**

  CONTROL_FILE_RECORD_KEEP_TIME
  DB_CREATE_ONLINE_LOG_DEST_*n*
  DB_RECOVERY_FILE_DEST
  DB_RECOVERY_FILE_DEST_SIZE
  DB_UNRECOVERABLE_SCN_TRACKING
  FAST_START_MTTR_TARGET
  LOG_BUFFER
  LOG_CHECKPOINT_INTERVAL
  LOG_CHECKPOINT_TIMEOUT
  LOG_CHECKPOINTS_TO_ALERT
  LOG_ARCHIVE_CONFIG
  LOG_ARCHIVE_DEST_*n*
  LOG_ARCHIVE_DEST_STATE_*n*
  LOG_ARCHIVE_DUPLEX_DEST
  LOG_ARCHIVE_FORMAT
  LOG_ARCHIVE_MAX_PROCESSES
  LOG_ARCHIVE_MIN_SUCCEED_DEST
  LOG_ARCHIVE_TRACE
  REDO_TRANSPORT_USER
  RECOVERY_PARALLELISM

- **Resource Manager**

  PROCESSOR_GROUP_NAME
  RESOURCE_LIMIT

```
RESOURCE_MANAGER_CPU_ALLOCATION
RESOURCE_MANAGER_PLAN
```

- **Security and Auditing**

```
AUDIT_FILE_DEST
AUDIT_SYS_OPERATIONS
AUDIT_SYSLOG_LEVEL
AUDIT_TRAIL
COMMIT_LOGGING
COMMIT_WAIT
DBFIPS_140
O7_DICTIONARY_ACCESSIBILITY
OS_AUTHENT_PREFIX
OS_ROLES
RDBMS_SERVER_DN
REMOTE_LOGIN_PASSWORDFILE
REMOTE_OS_AUTHENT
REMOTE_OS_ROLES
SEC_CASE_SENSITIVE_LOGON
SEC_MAX_FAILED_LOGIN_ATTEMPTS
SEC_PROTOCOL_ERROR_FURTHER_ACTION
SEC_PROTOCOL_ERROR_TRACE_ACTION
SEC_RETURN_SERVER_RELEASE_BANNER
SQL92_SECURITY
UNIFIED_AUDIT_SGA_QUEUE_SIZE
```

- **Sessions and Processes**

```
CPU_COUNT
PROCESSES
SESSIONS
THREADED_EXECUTION
USE_DEDICATED_BROKER
```

- **Shared Server Architecture**

```
CIRCUITS
DISPATCHERS
MAX_DISPATCHERS
MAX_SHARED_SERVERS
SHARED_SERVER_SESSIONS
SHARED_SERVERS
```

- **Standby Database**

```
ARCHIVE_LAG_TARGET
DB_FILE_NAME_CONVERT
DB_UNIQUE_NAME
```

DG_BROKER_CONFIG_FILE*n*
DG_BROKER_START
FAL_CLIENT
FAL_SERVER
LOG_FILE_NAME_CONVERT
STANDBY_FILE_MANAGEMENT

- **Temporary Sort Space**

  BITMAP_MERGE_AREA_SIZE
  CREATE_BITMAP_AREA_SIZE
  HASH_AREA_SIZE
  PGA_AGGREGATE_LIMIT
  PGA_AGGREGATE_TARGET
  SORT_AREA_RETAINED_SIZE
  SORT_AREA_SIZE
  WORKAREA_SIZE_POLICY

- **Transactions**

  COMMIT_LOGGING
  COMMIT_WAIT
  DB_LOST_WRITE_PROTECT
  DDL_LOCK_TIMEOUT
  DML_LOCKS
  FAST_START_PARALLEL_ROLLBACK
  GLOBAL_TXN_PROCESSES
  TRANSACTIONS

- **Undo Management**

  RESUMABLE_TIMEOUT
  ROLLBACK_SEGMENTS
  TEMP_UNDO_ENABLED
  TRANSACTIONS_PER_ROLLBACK_SEGMENT
  UNDO_MANAGEMENT
  UNDO_RETENTION
  UNDO_TABLESPACE

## Modifiable Parameters

Some initialization parameters can be modified using the ALTER SESSION or ALTER SYSTEM statements while an instance is running. Use the following syntax to modify initialization parameters:

```
ALTER SESSION SET parameter_name = value
ALTER SYSTEM SET parameter_name = value [DEFERRED]
```

Whenever a parameter is modified using the ALTER SYSTEM statement, the Oracle Database records the statement that modifies the parameter in the alert log.

The `ALTER SESSION` statement changes the value of the specified parameter for the duration of the session that invokes the statement. The value of the parameter does not change for other sessions in the instance. The value of the following initialization parameters can be changed with `ALTER SESSION`:

- ASM_DISKSTRING
- ASM_POWER_LIMIT
- COMMIT_LOGGING
- COMMIT_WAIT
- COMMIT_WRITE
- CREATE_STORED_OUTLINES
- CURSOR_BIND_CAPTURE_DESTINATION
- CURSOR_SHARING
- DB_BLOCK_CHECKING
- DB_CREATE_FILE_DEST
- DB_CREATE_ONLINE_LOG_DEST_*n*
- DB_FILE_MULTIBLOCK_READ_COUNT
- DB_FILE_NAME_CONVERT
- DB_INDEX_COMPRESSION_INHERITANCE
- DB_SECUREFILE
- DB_UNRECOVERABLE_SCN_TRACKING
- DDL_LOCK_TIMEOUT
- DEFERRED_SEGMENT_CREATION
- DST_UPGRADE_INSERT_CONV
- ENABLE_DDL_LOGGING
- GLOBAL_NAMES
- HASH_AREA_SIZE
- HEAT_MAP
- JAVA_JIT_ENABLED
- INMEMORY_CLAUSE_DEFAULT
- INMEMORY_QUERY
- LOG_ARCHIVE_DEST_*n*
- LOG_ARCHIVE_DEST_STATE_*n*
- LOG_ARCHIVE_MIN_SUCCEED_DEST
- MAX_DUMP_FILE_SIZE
- NLS_CALENDAR
- NLS_COMP
- NLS_CURRENCY
- NLS_DATE_FORMAT
- NLS_DATE_LANGUAGE
- NLS_DUAL_CURRENCY
- NLS_ISO_CURRENCY
- NLS_LANGUAGE
- NLS_LENGTH_SEMANTICS
- NLS_NCHAR_CONV_EXCP
- NLS_NUMERIC_CHARACTERS
- NLS_SORT
- NLS_TERRITORY
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT

```
OBJECT_CACHE_MAX_SIZE_PERCENT
OBJECT_CACHE_OPTIMAL_SIZE
OLAP_PAGE_POOL_SIZE
OPTIMIZER_ADAPTIVE_FEATURES
OPTIMIZER_ADAPTIVE_REPORTING_ONLY
OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES
OPTIMIZER_DYNAMIC_SAMPLING
OPTIMIZER_FEATURES_ENABLE
OPTIMIZER_INDEX_CACHING
OPTIMIZER_INDEX_COST_ADJ
OPTIMIZER_INMEMORY_AWARE
OPTIMIZER_MODE
OPTIMIZER_USE_INVISIBLE_INDEXES
OPTIMIZER_USE_PENDING_STATISTICS
OPTIMIZER_USE_SQL_PLAN_BASELINES
PARALLEL_DEGREE_LIMIT
PARALLEL_DEGREE_POLICY
PARALLEL_FORCE_LOCAL
PARALLEL_INSTANCE_GROUP
PARALLEL_IO_CAP_ENABLED
PARALLEL_MIN_PERCENT
PARALLEL_MIN_TIME_THRESHOLD
PDB_FILE_NAME_CONVERT
PLSCOPE_SETTINGS
PLSQL_CCFLAGS
PLSQL_CODE_TYPE
PLSQL_DEBUG
PLSQL_OPTIMIZE_LEVEL
PLSQL_V2_COMPATIBILITY
PLSQL_WARNINGS
QUERY_REWRITE_ENABLED
QUERY_REWRITE_INTEGRITY
RECYCLEBIN
REMOTE_DEPENDENCIES_MODE
RESULT_CACHE_MODE
RESULT_CACHE_REMOTE_EXPIRATION
RESUMABLE_TIMEOUT
SESSION_CACHED_CURSORS
SKIP_UNUSABLE_INDEXES
SMTP_OUT_SERVER
SORT_AREA_RETAINED_SIZE
SORT_AREA_SIZE
SPATIAL_VECTOR_ACCELERATION
SQL_TRACE
SQLTUNE_CATEGORY
STAR_TRANSFORMATION_ENABLED
STATISTICS_LEVEL
TEMP_UNDO_ENABLED
TIMED_OS_STATISTICS
```

```
TIMED_STATISTICS
TRACEFILE_IDENTIFIER
WORKAREA_SIZE_POLICY
XML_DB_EVENTS
```

The `ALTER SYSTEM` statement without the `DEFERRED` keyword modifies the global value of the parameter for all sessions in the instance, for the duration of the instance (until the database is shut down). The value of the following initialization parameters can be changed with `ALTER SYSTEM`:

```
AQ_TM_PROCESSES
AWR_SNAPSHOT_TIME_OFFSET
ARCHIVE_LAG_TARGET
ASM_DISKGROUPS
ASM_DISKSTRING
ASM_POWER_LIMIT
ASM_PREFERRED_READ_FAILURE_GROUPS
BACKGROUND_DUMP_DEST
CIRCUITS
COMMIT_LOGGING
COMMIT_WAIT
COMMIT_WRITE
CONNECTION_BROKERS
CONTROL_FILE_RECORD_KEEP_TIME
CONTROL_MANAGEMENT_PACK_ACCESS
CORE_DUMP_DEST
CPU_COUNT
CREATE_STORED_OUTLINES
CURSOR_BIND_CAPTURE_DESTINATION
CURSOR_SHARING
DB_nK_CACHE_SIZE
DB_BIG_TABLE_CACHE_PERCENT_TARGET
DB_BLOCK_CHECKING
DB_BLOCK_CHECKSUM
DB_CACHE_ADVICE
DB_CACHE_SIZE
DB_CREATE_FILE_DEST
DB_CREATE_ONLINE_LOG_DEST_n
DB_FILE_MULTIBLOCK_READ_COUNT
DB_FLASH_CACHE_FILE
DB_FLASH_CACHE_SIZE
DB_FLASHBACK_RETENTION_TARGET
DB_INDEX_COMPRESSION_INHERITANCE
DB_KEEP_CACHE_SIZE
DB_LOST_WRITE_PROTECT
DB_RECOVERY_FILE_DEST
DB_RECOVERY_FILE_DEST_SIZE
DB_RECYCLE_CACHE_SIZE
DB_SECUREFILE
DB_UNRECOVERABLE_SCN_TRACKING
```

```
DDL_LOCK_TIMEOUT
DEFERRED_SEGMENT_CREATION
DG_BROKER_CONFIG_FILEn
DG_BROKER_START
DIAGNOSTIC_DEST
DISPATCHERS
DST_UPGRADE_INSERT_CONV
ENABLE_DDL_LOGGING
ENABLE_GOLDENGATE_REPLICATION
FAL_CLIENT
FAL_SERVER
FAST_START_MTTR_TARGET
FAST_START_PARALLEL_ROLLBACK
FILE_MAPPING
FIXED_DATE
GLOBAL_NAMES
GLOBAL_TXN_PROCESSES
HEAT_MAP
HS_AUTOREGISTER
INMEMORY_CLAUSE_DEFAULT
INMEMORY_FORCE
INMEMORY_MAX_POPULATE_SERVERS
INMEMORY_QUERY
INMEMORY_SIZE
INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT
JAVA_JIT_ENABLED
JAVA_POOL_SIZE
JOB_QUEUE_PROCESSES
LARGE_POOL_SIZE
LDAP_DIRECTORY_ACCESS
LICENSE_MAX_SESSIONS
LICENSE_MAX_USERS
LICENSE_SESSIONS_WARNING
LISTENER_NETWORKS
LOCAL_LISTENER
LOG_ARCHIVE_CONFIG
LOG_ARCHIVE_DEST
LOG_ARCHIVE_DEST_n
LOG_ARCHIVE_DEST_STATE_n
LOG_ARCHIVE_DUPLEX_DEST
LOG_ARCHIVE_MAX_PROCESSES
LOG_ARCHIVE_MIN_SUCCEED_DEST
LOG_ARCHIVE_TRACE
LOG_CHECKPOINT_INTERVAL
LOG_CHECKPOINT_TIMEOUT
LOG_CHECKPOINTS_TO_ALERT
MAX_DISPATCHERS
MAX_DUMP_FILE_SIZE
MAX_SHARED_SERVERS
```

```
MEMORY_TARGET
NLS_LENGTH_SEMANTICS
NLS_NCHAR_CONV_EXCP
OPEN_CURSORS
OPTIMIZER_ADAPTIVE_FEATURES
OPTIMIZER_ADAPTIVE_REPORTING_ONLY
OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES
OPTIMIZER_DYNAMIC_SAMPLING
OPTIMIZER_FEATURES_ENABLE
OPTIMIZER_INDEX_CACHING
OPTIMIZER_INDEX_COST_ADJ
OPTIMIZER_INMEMORY_AWARE
OPTIMIZER_MODE
OPTIMIZER_SECURE_VIEW_MERGING
OPTIMIZER_USE_INVISIBLE_INDEXES
OPTIMIZER_USE_PENDING_STATISTICS
OPTIMIZER_USE_SQL_PLAN_BASELINES
PARALLEL_ADAPTIVE_MULTI_USER
PARALLEL_DEGREE_LIMIT
PARALLEL_DEGREE_POLICY
PARALLEL_FORCE_LOCAL
PARALLEL_INSTANCE_GROUP
PARALLEL_IO_CAP_ENABLED
PARALLEL_MAX_SERVERS
PARALLEL_MIN_SERVERS
PARALLEL_MIN_TIME_THRESHOLD
PARALLEL_SERVERS_TARGET
PARALLEL_THREADS_PER_CPU
PDB_CONVERT_FILE_NAME
PGA_AGGREGATE_LIMIT
PGA_AGGREGATE_TARGET
PLSCOPE_SETTINGS
PLSQL_CCFLAGS
PLSQL_CODE_TYPE
PLSQL_DEBUG
PLSQL_OPTIMIZE_LEVEL
PLSQL_V2_COMPATIBILITY
PLSQL_WARNINGS
QUERY_REWRITE_ENABLED
QUERY_REWRITE_INTEGRITY
REDO_TRANSPORT_USER
REMOTE_DEPENDENCIES_MODE
REMOTE_LISTENER
RESOURCE_LIMIT
RESOURCE_MANAGER_CPU_ALLOCATION
RESOURCE_MANAGER_PLAN
RESULT_CACHE_MAX_RESULT
RESULT_CACHE_MAX_SIZE
RESULT_CACHE_MODE
```

```
RESULT_CACHE_REMOTE_EXPIRATION
RESUMABLE_TIMEOUT
SEC_CASE_SENSITIVE_LOGON
SEC_PROTOCOL_ERROR_FURTHER_ACTION
SEC_PROTOCOL_ERROR_TRACE_ACTION
SERVICE_NAMES
SGA_TARGET
SHARED_POOL_SIZE
SHARED_SERVER_SESSIONS
SHARED_SERVERS
SKIP_UNUSABLE_INDEXES
SMTP_OUT_SERVER
SPATIAL_VECTOR_ACCELERATION
SPFILE
SQL_TRACE
SQLTUNE_CATEGORY
STANDBY_ARCHIVE_DEST
STANDBY_FILE_MANAGEMENT
STAR_TRANSFORMATION_ENABLED
STATISTICS_LEVEL
STREAMS_POOL_SIZE
TEMP_UNDO_ENABLED
THREAD
TIMED_OS_STATISTICS
TIMED_STATISTICS
TRACE_ENABLED
UNDO_RETENTION
UNDO_TABLESPACE
USE_DEDICATED_BROKER
USER_DUMP_DEST
WORKAREA_SIZE_POLICY
XML_DB_EVENTS
```

The `ALTER SYSTEM ... DEFERRED` statement does not modify the global value of the parameter for existing sessions, but the value will be modified for future sessions that connect to the database. The value of the following initialization parameters can be changed with `ALTER SYSTEM ... DEFERRED`:

```
AUDIT_FILE_DEST
BACKUP_TAPE_IO_SLAVES
OBJECT_CACHE_MAX_SIZE_PERCENT
OBJECT_CACHE_OPTIMAL_SIZE
OLAP_PAGE_POOL_SIZE
RECYCLEBIN
SORT_AREA_RETAINED_SIZE
SORT_AREA_SIZE
```

## Displaying Current Parameter Values

To see the current settings for initialization parameters, use the following SQL*Plus command:

```
SQL> SHOW PARAMETERS
```

This command displays all parameters in alphabetical order, along with their current values.

Enter the following text string to display all parameters having `BLOCK` in their names:

```
SQL> SHOW PARAMETERS BLOCK
```

You can use the `SPOOL` command to write the output to a file.

## Parameters You Should Not Specify in the Parameter File

You should not specify the following two types of parameters in your parameter files:

- Parameters that you never alter unless instructed to do so by Oracle to resolve a problem
- Derived parameters, which normally do not need altering because their values are calculated automatically by the Oracle database server

## When Parameters Are Set Incorrectly

Some parameters have a minimum setting below which an Oracle instance will not start. For other parameters, setting the value too low or too high may cause Oracle to perform badly, but it will still run. Also, Oracle may convert some values outside the acceptable range to usable levels.

If a parameter value is too low or too high, or you have reached the maximum for some resource, then Oracle returns an error. Frequently, you can wait a short while and retry the operation when the system is not as busy. If a message occurs repeatedly, then you should shut down the instance, adjust the relevant parameter, and restart the instance.

# Reading the Parameter Descriptions

The parameter descriptions in this chapter adhere to the following format.

### PARAMETER_NAME

| Property | Description |
|---|---|
| **Parameter type** | Specifies the type of the parameter:<br>• A Boolean parameter accepts either `true` or `false` as its value.<br>• A string parameter accepts any sequence of characters as its value, subject to the syntax for the parameter.<br>• An integer parameter accepts a 4-byte value that can range from 0 to $2^{32}$ - 1.<br>• A parameter file parameter accepts an initialization parameter file specification as its value.<br>• A **big integer** parameter accepts an 8-byte value that can range from 0 to $2^{64}$ - 1. You specify a value for a big integer as an integer |

| Property | Description |
|---|---|
| | together with an optional modifier such as K, M, or G, which respectively denotes kilobytes, megabytes, or gigabytes.<br>For example, 1000, 100 KB, 50 MB and 2 GB are valid specifications for big integers. |
| **Syntax** | For string and big integer parameters, specifies the valid syntax for specifying the parameter. |
| **Default value** | Specifies the value this parameter assumes if not explicitly specified. |
| **Modifiable** | Specifies whether the parameter can be changed for the current session (by an `ALTER SESSION` statement) or for all sessions in the current instance (by an `ALTER SYSTEM` statement):<br>• `ALTER SESSION` overrides the instance-wide setting of the parameter for the current session only. You can restore the instance-wide setting for that session only by issuing another `ALTER SESSION` statement.<br>• `ALTER SYSTEM` can be used to change the value in the server parameter file (SPFILE) of any initialization parameter. Such a change takes effect only in subsequent instances. The parameter descriptions indicate only those parameters that can be modified for the current instance. |
| **Range of values** | Specifies the valid range of values that this parameter can assume, shown as a minimum and maximum value. Not applicable to all parameters. |
| **Basic** | Specifies whether the parameter is a basic initialization parameter or not |
| **Oracle RAC** | Specifies how the values for this parameter must be specified for multiple instances in an Oracle Real Application Clusters environment. Not applicable to all parameters. |

For each parameter, paragraphs following these details further describe the parameter and the effects of different settings.

## Initialization Parameter Descriptions

The remainder of this chapter describes the initialization parameters in alphabetical order. Initialization parameter values apply to the entire database, not to an individual user, unless otherwise specified.

**Note:**

Parameters that have become obsolete are not documented.

**See Also:**

- *Oracle Database Upgrade Guide* for information about obsolete parameters
- Your system release bulletins or other operating system-specific Oracle documentation

# ACTIVE_INSTANCE_COUNT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 1 |
| **Modifiable** | No |
| **Range of values** | 1 or >= the number of instances in the cluster. (Values other than 1 have no effect on the active or standby status of any instances.) |
| **Basic** | No |
| **Oracle RAC** | You must set this parameter for every instance, and multiple instances must have the same value. |

**Note:**

The `ACTIVE_INSTANCE_COUNT` parameter is deprecated. It is retained for backward compatibility only.

`ACTIVE_INSTANCE_COUNT` enables you to designate one instance in a two-instance cluster as the primary instance and the other instance as the secondary instance. This parameter has no functionality in a cluster with more than two instances.

When you set this parameter to 1, the first instance you start up becomes the primary instance and accepts client connections. The second instance starts up as a secondary instance and can accept client connections only if the first instance fails. In such an event, the secondary instance becomes the primary instance.

When the failed instance can once again be started up, it starts up as the secondary instance, and will not accept client connections unless the current primary instance fails.

**Note:**

This parameter functions only in a cluster with only two instances.

# AQ_TM_PROCESSES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `10485760` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `0` to `40` |
| **Basic** | No |

AQ_TM_PROCESSES controls time monitoring on queue messages and controls processing of messages with delay and expiration properties specified.You do not need to specify a value for this parameter because Oracle Database automatically determines the number of processes and autotunes them, as necessary. Therefore, Oracle highly recommends that you leave the AQ_TM_PROCESSES parameter unspecified and let the system autotune.

The default value for AQ_TM_PROCESSES is used if the client does not explicitly set a value for the parameter in the init.ora file or using the ALTER SYSTEM command.

**Note:**

If you want to disable the Queue Monitor Coordinator, then you must set AQ_TM_PROCESSES to 0 in your parameter file. Oracle strongly recommends that you do NOT set AQ_TM_PROCESSES to 0. If you are using Oracle Streams, then setting this parameter to zero (which Oracle Database respects no matter what) can cause serious problems.

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about this parameter

# ARCHIVE_LAG_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 (disabled) |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 or any integer in [60, 7200] |
| **Basic** | No |
| **Oracle RAC** | Multiple instances should use the same value |

ARCHIVE_LAG_TARGET forces a log switch after the specified amount of time elapses.

A 0 value disables the time-based thread advance feature; otherwise, the value represents the number of seconds. Values larger than 7200 seconds are not of much use in maintaining a reasonable lag in the standby database. The typical, or recommended value is 1800 (30 minutes). Extremely low values can result in frequent log switches, which could degrade performance; such values can also make the archiver process too busy to archive the continuously generated logs.

**See Also:**

*Oracle Database Administrator's Guide* for more information about setting this parameter

# ASM_DISKGROUPS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `ASM_DISKGROUPS = diskgroup [, diskgroup ] ...` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Comma-separated list of strings, up to 30 characters |
| **Oracle RAC** | Multiple instances can have different values. |

**Note:**

This parameter may only be specified in an Oracle Automatic Storage Management (Oracle ASM) instance.

`ASM_DISKGROUPS` specifies a list of disk group names that an Oracle ASM instance mounts when the `ALTER DISKGROUP ALL MOUNT` statement is issued. The Oracle ASM instance startup process executes `ALTER DISKGROUP ALL MOUNT` unless the `NOMOUNT` startup option is specified.

Oracle ASM automatically adds a disk group to this parameter when the disk group is successfully created or mounted, and automatically removes a disk group from this parameter when the disk group is dropped or dismounted.

Issuing the `ALTER DISKGROUP...ALL MOUNT` or `ALTER DISKGROUP...ALL DISMOUNT` command does not affect the value of this parameter.

### Supporting Up to 511 Disk Groups for an Oracle ASM Instance

In Oracle Database 12*c* Release 1 or later, Oracle ASM configurations support up to 511 disk groups. Oracle ASM configurations with Oracle Database releases before 12*c* Release 1 can only support up to 63 disk groups.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for more information about and examples of using this parameter

# ASM_DISKSTRING

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `ASM_DISKSTRING = discovery_string [, discovery_string ] ...` |
| **Default value** | The null string; Oracle Automatic Storage Management discovery finds all disks in an operating system-specific location to which the |

| Property | Description |
|---|---|
| | Oracle Automatic Storage Management instance has read/write access. |
| **Modifiable** | `ALTER SESSION, ALTER SYSTEM` |
| **Oracle RAC** | Multiple instances can have different values. Different nodes might see the same disks under different names; however, each instance must be able to use its `ASM_DISKSTRING` to discover the same physical media as the other nodes in the cluster. |

**Note:**

This parameter may only be specified in an Oracle Automatic Storage Management (Oracle ASM) instance.

`ASM_DISKSTRING` specifies an operating system-dependent value used by Oracle ASM to limit the set of disks considered for discovery. When a new disk is added to a disk group, each Oracle ASM instance that has the disk group mounted must be able to discover the new disk using the value of `ASM_DISKSTRING`.

In most cases, the default value will be sufficient. Using a more restrictive value may reduce the time required for Oracle ASM to perform discovery, and thus improve disk group mount time or the time for adding a disk to a disk group. A "?" at the beginning of the string gets expanded to the Oracle home directory. Depending on the operating system, wildcard characters can be used. It may be necessary to dynamically change `ASM_DISKSTRING` before adding a disk so that the new disk will be discovered.

An attempt to dynamically modify `ASM_DISKSTRING` will be rejected and the old value retained if the new value cannot be used to discover a disk that is in a disk group that is already mounted.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for more information and examples of using this parameter

# ASM_POWER_LIMIT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `0` to `11`[1] |
| **Oracle RAC** | Multiple instances can have different values. |

[1] Beginning with Oracle Database 11*g* Release 2 (11.2.0.2), if the `COMPATIBLE.ASM` disk group attribute is set to 11.2.0.2 or higher, then the range of values is 0 to 1024.

**Note:**

This parameter may only be specified in an Oracle Automatic Storage Management (Oracle ASM) instance.

ASM_POWER_LIMIT specifies the maximum power on an Oracle ASM instance for disk rebalancing. The higher the limit, the faster rebalancing will complete. Lower values will take longer, but consume fewer processing and I/O resources.

If the POWER clause of a rebalance operation is not specified, then the default power will be the value of ASM_POWER_LIMIT.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for more information about using this parameter

# ASM_PREFERRED_READ_FAILURE_GROUPS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | ASM_PREFERRED_READ_FAILURE_GROUPS = <diskgroup_name>.<failure_group_name>, ... |
| **Default value** | NULL |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |
| **Oracle RAC** | The value is different on different nodes |

ASM_PREFERRED_READ_FAILURE_GROUPS specifies the failure groups that contain preferred read disks. Preferred disks are instance specific. This parameter is only valid in Oracle Automatic Storage Management (Oracle ASM) instances.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for more information about this parameter

# AUDIT_FILE_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | AUDIT_FILE_DEST = '*directory*' |
| **Default value** | The first default value is: *ORACLE_BASE*/admin/*ORACLE_SID*/adump |

| Property | Description |
|---|---|
| | The second default value, which is used if the first default value does not exist or is unusable, is:<br>*ORACLE_HOME*/rdbms/audit<br>Both of these default values are for UNIX systems. Other platforms may have different defaults.<br>In a multitenant container database (CDB), both of these default values will be appended with the GUID of the pluggable database (PDB) to store audit records that belong to the PDB. For example, if the PDB's GUID is 03E1F908EE04252CE053B280E80AAAA3, the first default directory will be:<br>*ORACLE_BASE*/admin/*ORACLE_SID*/adump/03E1F908EE04252CE053B280E80AAAA3<br>You can use the V$CONTAINERS view to query a PDB's GUID. |
| **Modifiable** | ALTER SYSTEM ... DEFERRED |
| **Basic** | No |

**Note:**

In an Oracle database that has migrated to unified auditing, the setting of this parameter has no effect.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

AUDIT_FILE_DEST specifies the operating system directory into which the audit trail is written when the AUDIT_TRAIL initialization parameter is set to **os**, **xml**, or **xml,extended**. The audit records will be written in XML format if the AUDIT_TRAIL initialization parameter is set to **xml** or **xml, extended**. It is also the location to which mandatory auditing information is written and, if so specified by the AUDIT_SYS_OPERATIONS initialization parameter, audit records for user SYS.

In a multitenant container database (CDB), the scope of the settings for this initialization parameter is the CDB. Although the audit trail is provided per pluggable database (PDB) in a CDB, this initialization parameter cannot be configured for individual PDBs.

**See Also:**

- *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for more information about CDBs and PDBs
- "V$CONTAINERS" for more information about the V$CONTAINERS view

# AUDIT_SYS_OPERATIONS

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | TRUE |
| **Modifiable** | No |
| **Range of values** | TRUE \| FALSE |
| **Basic** | No |

**Note:**

In an Oracle database that has migrated to unified auditing, the setting of this parameter has no effect.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

AUDIT_SYS_OPERATIONS enables or disables the auditing of top-level operations, which are SQL statements directly issued by users when connecting with the SYSASM, SYSBACKUP, SYSDBA, SYSDG, SYSKM, or SYSOPER privileges. (SQL statements run from within PL/SQL procedures or functions are not considered top-level.) The audit records are written to the operating system's audit trail. The audit records will be written in XML format if the AUDIT_TRAIL initialization parameter is set to xml or xml, extended.

On UNIX platforms, if the AUDIT_SYSLOG_LEVEL parameter has also been set, then it overrides the AUDIT_TRAIL parameter and SYS audit records are written to the system audit log using the SYSLOG utility.

In a CDB, the scope of the settings for this initialization parameter is the CDB. Although the audit trail is provided per PDB in a CDB, this initialization parameter cannot be configured for individual PDBs.

# AUDIT_SYSLOG_LEVEL

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | AUDIT_SYSLOG_LEVEL = *'facility_clause.priority_clause'*<br><br>**facility_clause::=**<br>{ USER \| LOCAL[0 \| 1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7] \| SYSLOG \| DAEMON \| KERN \| MAIL \| AUTH \| LPR \| NEWS \| UUCP \| CRON }<br><br>**priority_clause::=**<br>{ NOTICE \| INFO \| DEBUG \| WARNING \| ERR \| CRIT \| ALERT \| EMERG } |

| Property | Description |
|---|---|
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Basic** | No |

**Examples**

```
AUDIT_SYSLOG_LEVEL = 'KERN.EMERG';
AUDIT_SYSLOG_LEVEL = 'LOCAL1.WARNING';
```

**Note:**

In an Oracle database that has migrated to unified auditing, the setting of this parameter has no effect.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

`AUDIT_SYSLOG_LEVEL` allows `SYS` and standard `OS` audit records to be written to the system audit log using the `SYSLOG` utility.

If you use this parameter, it is best to assign a file corresponding to every combination of facility and priority (especially `KERN.EMERG`) in `syslog.conf` . Sometimes these are assigned to print to the console in the default `syslog.conf` file. This can become annoying and will be useless as audit logs. Also, if you use this parameter, it is best to set the maximum length of syslog messages in the system to 512 bytes.

**Note:** Audit records written to the system audit log could get truncated to 512 bytes, and different parts of the same audit record may not be joined to get the original complete audit record.

**See Also:**

*Oracle Database Security Guide* for information about configuring syslog auditing

If `AUDIT_SYSLOG_LEVEL` is set and `SYS` auditing is enabled (`AUDIT_SYS_OPERATIONS` = `TRUE`), then `SYS` audit records are written to the system audit log.If `AUDIT_SYSLOG_LEVEL` is set and standard audit records are being sent to the operating system (`AUDIT_TRAIL` = `os`), then standard audit records are written to the system audit log.

In a CDB, the scope of the settings for this initialization parameter is the CDB. Although the audit trail is provided per PDB in a CDB, this initialization parameter cannot be configured for individual PDBs.

# AUDIT_TRAIL

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `AUDIT_TRAIL = { none \| os \| db [, extended] \| xml [, extended] }` |
| **Default value** | `none` |
| **Modifiable** | No |
| **Basic** | No |

**Note:**

In an Oracle database that has migrated to unified auditing, the setting of this parameter has no effect.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

`AUDIT_TRAIL` enables or disables database auditing.

### Values

- `none`

  Disables standard auditing. This value is the default if the `AUDIT_TRAIL` parameter was not set in the initialization parameter file or if you created the database using a method other than Database Configuration Assistant. If you created the database using Database Configuration Assistant, then the default is `db`.

- `os`

  Directs all audit records to an operating system file. Oracle recommends that you use the `os` setting, particularly if you are using an ultra-secure database configuration.

- `db`

  Directs audit records to the database audit trail (the `SYS.AUD$` table), except for records that are always written to the operating system audit trail. Use this setting for a general database for manageability.

  If the database was started in read-only mode with `AUDIT_TRAIL` set to `db`, then Oracle Database internally sets `AUDIT_TRAIL` to `os`. Check the alert log for details.

- `db, extended`

  Performs all actions of `AUDIT_TRAIL=db`, and also populates the SQL bind and SQL text CLOB-type columns of the `SYS.AUD$` table, when available. These two columns are populated only when this parameter is specified.

If the database was started in read-only mode with AUDIT_TRAIL set to db, extended, then Oracle Database internally sets AUDIT_TRAIL to os. Check the alert log for details.

- xml

  Writes to the operating system audit record file in XML format. Records all elements of the AuditRecord node except Sql_Text and Sql_Bind to the operating system XML audit file.

- xml, extended

  Performs all actions of AUDIT_TRAIL=xml, and includes SQL text and SQL bind information in the audit trail.

You can use the SQL AUDIT statement to set auditing options regardless of the setting of this parameter.

In a CDB, the scope of the settings for this initialization parameter is the CDB. Although the audit trail is provided per PDB in a CDB, this initialization parameter cannot be configured for individual PDBs.

### Examples

The following statement sets the db, extended value for the AUDIT_TRAIL parameter. The new value takes effect after the database is restarted.

```
SQL> alter system set AUDIT_TRAIL=db, extended scope=spfile;

System altered.

SQL>
```

The following statement sets the xml, extended value for the AUDIT_TRAIL parameter. The new value takes effect after the database is restarted.

```
SQL> alter system set AUDIT_TRAIL=xml, extended scope=spfile;

System altered.

SQL>
```

The following statement sets the db value for the AUDIT_TRAIL parameter. The new value takes effect after the database is restarted.

```
SQL> alter system set AUDIT_TRAIL=db scope=spfile;

System altered.

SQL>
```

**See Also:**

- *Oracle Database Security Guide* for information about configuring unified audit policies
- *Oracle Database Upgrade Guide* to learn more about traditional non-unified auditing

# AWR_SNAPSHOT_TIME_OFFSET

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 200 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 - 3599, or the special value 1000000 |
| **Basic** | No |
| **Oracle RAC** | Multiple instances should use the same value |

AWR_SNAPSHOT_TIME_OFFSET specifies an offset for the Automatic Workload Repository (AWR) snapshot start time.

AWR snapshots normally start at the top of the hour (12:00, 1:00, 2:00, and so on). This parameter allows DBAs to specify an offset for the AWR snapshot start time.

This is a useful parameter to avoid CPU spikes from multiple instances all starting their AWR snapshots at the same time. If you have a large system with many instances on it (like many Exadata installations), and you are experiencing such CPU spikes, this parameter can be very useful.

The parameter is specified in seconds. Normally, you set it to a value less than 3600. If you set the special value 1000000 (1,000,000), you get an automatic mode, in which the offset is based on the database name.

The automatic mode is an effective way of getting a reasonable distribution of offset times when you have a very large number of instances running on the same node.

# BACKGROUND_CORE_DUMP

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | BACKGROUND_CORE_DUMP = { partial \| full } |
| **Default value** | partial |
| **Modifiable** | No |
| **Basic** | No |

BACKGROUND_CORE_DUMP specifies whether Oracle includes the SGA in the core file for Oracle background processes.

### Values

- partial

  Oracle does not include the SGA in the core dump.

- full

  Oracle includes the SGA in the core dump.

**See Also:**

"SHADOW_CORE_DUMP"

# BACKGROUND_DUMP_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `BACKGROUND_DUMP_DEST = { pathname \| directory }` |
| **Default value** | Operating system-dependent |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Any valid local path, directory, or disk |
| **Basic** | No |

**Note:**

The `BACKGROUND_DUMP_DEST` parameter is deprecated in Oracle Database 12*c* Release 1 (12.1.0.1).

`BACKGROUND_DUMP_DEST` specifies the pathname (directory or disc) where debugging trace files for the background processes (LGWR, DBW*n*, and so on) are written during Oracle operations.

An **alert log** in the directory specified by `BACKGROUND_DUMP_DEST` logs significant database events and messages. Anything that affects the database instance or global database is recorded here. The alert log is a normal text file. Its file name is operating system-dependent. For platforms that support multiple instances, it takes the form `alert_sid.log`, where *sid* is the system identifier. This file grows slowly, but without limit, so you might want to delete it periodically. You can delete the file even when the database is running.

**Note:**

This parameter is ignored by the diagnosability infrastructure introduced in Oracle Database 11g Release 1 (11.1), which places trace and core files in a location controlled by the `DIAGNOSTIC_DEST` initialization parameter.

**See Also:**

- *Oracle Database Administrator's Guide* for more information on the DIAGNOSTIC_DEST initialization parameter
- "USER_DUMP_DEST" for information on setting a destination for server process trace files

# BACKUP_TAPE_IO_SLAVES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SYSTEM ... DEFERRED` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`BACKUP_TAPE_IO_SLAVES` specifies whether I/O server processes (also called **slaves**) are used by Recovery Manager to back up, copy, or restore data to tape. When the value is set to `true`, Oracle uses an I/O server process to write to or read from a tape device. When the value is set to `false` (the default), Oracle does not use I/O server process for backups. Instead, the shadow process engaged in the backup accesses the tape device.

**Note:**

You cannot perform duplexed backups unless you enable this parameter. Otherwise, Oracle returns an error. When this parameter is enabled, Recovery Manager will configure as many server processes as needed for the number of backup copies requested.

**See Also:**

- *Oracle Database Backup and Recovery User's Guide* for more information on duplexed backups
- "DBWR_IO_SLAVES"

# BITMAP_MERGE_AREA_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `8388608` |
| **Modifiable** | No |
| **Range of values** | Operating system-dependent |
| **Basic** | No |

**Note:**

Oracle does not recommend using the BITMAP_MERGE_AREA_SIZE parameter unless the instance is configured with the shared server option. Oracle recommends that you enable automatic sizing of SQL working areas by setting PGA_AGGREGATE_TARGET instead. BITMAP_MERGE_AREA_SIZE is retained for backward compatibility.

BITMAP_MERGE_AREA_SIZE is relevant only for systems containing bitmap indexes. It specifies the amount of memory Oracle uses to merge bitmaps retrieved from a range scan of the index. A larger value usually improves performance, because the bitmap segments must be sorted before being merged into a single bitmap.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information on using bitmap indexes for performance

# BLANK_TRIMMING

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | No |
| **Range of values** | true \| false |
| **Basic** | No |

BLANK_TRIMMING specifies the data assignment semantics of character datatypes.

### Values

- TRUE

  Allows the data assignment of a source character string or variable to a destination character column or variable even though the source length is longer than the destination length. In this case, however, the additional length over the destination length must be all blanks, else an exception condition is raised. This value complies with the semantics of SQL-92 Transitional Level and above.

- FALSE

  Disallows the data assignment if the source length is longer than the destination length and reverts to SQL92 Entry Level semantics.

**See Also:**

*Oracle Database Globalization Support Guide* for more information on how using this parameter can help prevent data truncation issues during character set migration

# CIRCUITS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `4294967295` |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`CIRCUITS` specifies the total number of virtual circuits that are available for inbound and outbound network sessions. It is one of several parameters that contribute to the total SGA requirements of an instance.

You should not specify a value for this parameter unless you want to limit the number of virtual circuits.

**See Also:**

- *Oracle Database Concepts* for more information on memory structures
- *Oracle Database Concepts* for more information on processes

# CLIENT_RESULT_CACHE_LAG

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `CLIENT_RESULT_CACHE_LAG = integer` |
| **Default value** | `0` |
| **Modifiable** | No |
| **Range of values** | `0` to operating system-dependent |
| **Basic** | No |

`CLIENT_RESULT_CACHE_LAG` specifies the maximum time (in milliseconds) since the last round trip to the server, before which the OCI client query execute makes a round trip to get any database changes related to the queries cached on the client.

**See Also:**

*Oracle Call Interface Programmer's Guide* for more information about the client query cache feature

# CLIENT_RESULT_CACHE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `CLIENT_RESULT_CACHE_SIZE = integer [K \| M \| G]` |
| **Default value** | `1` |
| **Modifiable** | No |
| **Range of values** | `0` to operating system-dependent |
| **Basic** | No |

`CLIENT_RESULT_CACHE_SIZE` specifies the maximum size of the client per-process result set cache (in bytes). All OCI client processes inherit this maximum size. Setting a nonzero value enables the client query cache feature. This can be overridden by the client configuration parameter `OCI_RESULT_CACHE_MAX_SIZE`.

**See Also:**

*Oracle Call Interface Programmer's Guide* for more information about the client query cache feature

# CLONEDB

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | The same value should be set for all instances. |

`CLONEDB` should be set on Direct NFS Client CloneDB databases. When this parameter is set, the CloneDB database uses the database backup as the backing store for the datafiles.

**See Also:**

See *Oracle Database Administrator's Guide* for more information about cloning databases on network attached storage (NAS).

# CLUSTER_DATABASE

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | Yes |
| **Oracle RAC** | For all instances, the value must be set to `true`. |

`CLUSTER_DATABASE` is an Oracle RAC parameter that specifies whether Oracle RAC is enabled.

**See Also:**

*Oracle Real Application Clusters Administration and Deployment Guide* for an introduction to Oracle RAC

# CLUSTER_DATABASE_INSTANCES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | If `CLUSTER_DATABASE` is set to `false`, then `1`<br>If `CLUSTER_DATABASE` is set to `true`:<br>• If Oracle ASM, then 4<br>• Otherwise, the number of configured Oracle RAC instances |
| **Modifiable** | No |
| **Range of values** | Any nonzero value |
| **Basic** | No |
| **Oracle RAC** | Multiple instances should have the same value. |

`CLUSTER_DATABASE_INSTANCES` is an Oracle RAC parameter that specifies the number of instances that are configured as part of the cluster database. You must set this parameter for every instance. Normally you should set this parameter to the number of instances in your Oracle RAC environment. A proper setting for this parameter can improve memory use.

Oracle uses the value of this parameter to compute the default value of the `LARGE_POOL_SIZE` parameter when the `PARALLEL_AUTOMATIC_TUNING` parameter is set to `true`. Note that the `PARALLEL_AUTOMATIC_TUNING` parameter has been deprecated.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about parallel execution
- *Oracle Real Application Clusters Administration and Deployment Guide* for information on Real Application Clusters

# CLUSTER_INTERCONNECTS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `CLUSTER_INTERCONNECTS = ifn [: ifn ] ...` |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | One or more IP addresses, separated by colons |
| **Basic** | No |

`CLUSTER_INTERCONNECTS` can be used in Oracle Real Application Clusters environments to indicate cluster interconnects available for use for the database traffic. Use this parameter to override the default interconnect configured for the database traffic, which is stored in the cluster registry. This procedure also may be useful with Data Warehouse systems that have reduced availability requirements and high interconnect bandwidth demands.

`CLUSTER_INTERCONNECTS` specifically overrides the following:

- Network classifications stored by `oifcfg` in the OCR.
- The default interconnect chosen by Oracle.

If you want to load-balance the interconnect, then Oracle recommends that you use link-bonding at the operating system level, even if you have two databases on the same server, so that multiple interconnects use the same address. Note that multiple private addresses provide load balancing, but do not provide failover unless bonded. If you specify multiple addresses in init.ora using `CLUSTER_INTERCONNECTS`, instead of bonding multiple addresses at the operating system level, then typically availability is reduced, because each network interface card failure will take down that instance.

Refer to your vendor documentation for information about bonding interfaces. Some vendor bonding architectures may require the use of this parameter.

If you have multiple database instances on Oracle Real Application Clusters nodes and want to use a specific interface for each instance, then you can set the `CLUSTER_INTERCONNECTS` initialization parameter to the IP address for each database instance. For example:

```
hr1.init.ora.cluster_interconnects="192.0.2.111"
oltp3.init.ora.cluster_interconnects="192.0.2.112"
```

If the Oracle RAC interconnect is configured to run on a different interface than the Oracle Clusterware interconnect, then this configuration can cause reduced availability, as failovers or instance evictions can be delayed if the Oracle RAC interconnect fails while the Oracle Clusterware NIC remains up.

**See Also:**

*Oracle Real Application Clusters Administration and Deployment Guide* for additional information about using `CLUSTER_INTERCONNECTS`

## COMMIT_LOGGING

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `COMMIT_LOGGING = { IMMEDIATE \| BATCH }` |
| **Default value** | There is no default value. |
| **Modifiable** | Yes (at both session-level and system-level) |
| **Basic** | No |
| **Oracle RAC** | Each instance may have its own setting |

`COMMIT_LOGGING` is an advanced parameter used to control how redo is batched by Log Writer.

If `COMMIT_LOGGING` is altered after setting `COMMIT_WAIT` to `FORCE_WAIT`, then the `FORCE_WAIT` option is no longer valid.

## COMMIT_POINT_STRENGTH

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `50` |
| **Modifiable** | No |
| **Range of values** | `0` to `255` |
| **Basic** | No |

`COMMIT_POINT_STRENGTH` is relevant only in distributed database systems. It specifies a value that determines the **commit point site** in a distributed transaction. The node in the transaction with the highest value for `COMMIT_POINT_STRENGTH` will be the commit point site.

The commit point site of a database should reflect the amount of critical shared data in the database. For example, a database on a mainframe computer typically shares more data among users than one on a personal computer. Therefore, `COMMIT_POINT_STRENGTH` should be set to a higher value for the mainframe computer.

The commit point site stores information about the status of transactions. Other computers in a distributed transaction require this information during Oracle's two-phase commit, so it is desirable to have machines that are always available as commit point sites. Therefore, set COMMIT_POINT_STRENGTH to a higher value on your more available machines.

**See Also:**

- *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for information on two-phase commit
- Your operating system-specific Oracle documentation for the default value of this parameter

# COMMIT_WAIT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | COMMIT_WAIT = { NOWAIT \| WAIT \| FORCE_WAIT } |
| **Default value** | There is no default value. |
| **Modifiable** | Yes (at both session-level and system-level) |
| **Basic** | No |
| **Oracle RAC** | Each instance may have its own setting |

COMMIT_WAIT is an advanced parameter used to control when the redo for a commit is flushed to the redo logs.

Be aware that the NOWAIT option can cause a failure that occurs after the database receives the commit message, but before the redo log records are written. This can falsely indicate to a transaction that its changes are persistent. Also, it can violate the durability of ACID (Atomicity, Consistency, Isolation, Durability) transactions if the database shuts down unexpectedly.

If the parameter is set to FORCE_WAIT, the default behavior (immediate flushing of the redo log buffer with wait) is used. If this is a system setting, the session level and transaction level options will be ignored. If this is a session level setting, the transaction level options will be ignored. If COMMIT_WAIT is altered after it has been set to FORCE_WAIT, then the FORCE_WAIT option is no longer valid.

# COMMIT_WRITE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | COMMIT_WRITE = '{IMMEDIATE \| BATCH},{WAIT \|NOWAIT}' |
| **Default value** | If this parameter is not explicitly specified, then database commit behavior defaults to writing commit records to disk before control is returned to the client. |

| Property | Description |
|---|---|
| | If only IMMEDIATE or BATCH is specified, but not WAIT or NOWAIT, then WAIT mode is assumed.<br>If only WAIT or NOWAIT is specified, but not IMMEDIATE or BATCH, then IMMEDIATE mode is assumed.<br>Be aware that the NOWAIT option can cause a failure that occurs after the database receives the commit message, but before the redo log records are written. This can falsely indicate to a transaction that its changes are persistent. Also, it can violate the durability of ACID (Atomicity, Consistency, Isolation, Durability) transactions if the database shuts down unexpectedly. |
| **Modifiable** | Yes (at both session-level and system-level). Values supplied for COMMIT_WRITE in an ALTER SYSTEM or ALTER SESSION statement must be separated by a comma. |
| **Range of values** | Single-quoted, comma-separated list of either IMMEDIATE or BATCH, and either WAIT or NOWAIT. |
| **Basic** | No |
| **Oracle RAC** | Each instance may have its own setting |

COMMIT_WRITE is an advanced parameter used to control how redo for transaction commits is written to the redo logs. The IMMEDIATE and BATCH options control how redo is batched by Log Writer. The WAIT and NOWAIT options control when the redo for a commit is flushed to the redo logs.

**Note:**

The COMMIT_WRITE parameter is deprecated. It is retained for backward compatibility only. It is replaced by the COMMIT_LOGGING and COMMIT_WAIT parameters.

# COMMON_USER_PREFIX

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | COMMON_USER_PREFIX = prefix |
| **Default value** | NULL |
| **Modifiable** | No |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must have the same value |

COMMON_USER_PREFIX specifies a prefix that the names of common users, roles, and profiles in a multitenant container database (CDB) must start with. Names of local users, roles, and profiles must not start with the same prefix. A case-insensitive comparison of the prefixes for user, role, and profile names to the value of this parameter is done.

**Note:**

Regardless of the value of the COMMON_USER_PREFIX parameter, the names of local users, roles, and profiles cannot start with C##.

If COMMON_USER_PREFIX is set to an empty string, Oracle will *not* enforce any restrictions on the names of common or local users, roles, and profiles.

**Note:**

Setting this parameter to an empty string will result in no restrictions being placed on names of common and local users, roles, and profiles, which could lead to conflicts between the names of local and common users, roles, and profiles when a PDB is plugged into a different CDB, or when opening a PDB that was closed when a common user was created.

When a value is set for COMMON_USER_PREFIX, Oracle will require that the names of common users, roles, and profiles start with the string assigned to this parameter.

The names of users, roles, and profiles created using this prefix must be valid names.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- *Oracle Database SQL Language Reference* for information about valid user names
- *Oracle Database SQL Language Reference* for information about valid role names
- *Oracle Database SQL Language Reference* for information about valid profile names
- *Oracle Database Concepts* for more information about the multitenant architecture

# COMPATIBLE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | COMPATIBLE = *release_number* |
| **Default value** | 12.0.0 |
| **Modifiable** | No |

| Property | Description |
|---|---|
| **Range of values** | `11.0.0` to default release |
| **Basic** | Yes |
| **Oracle RAC** | Multiple instances must have the same value. |

`COMPATIBLE` enables you to use a new release of Oracle, while at the same time guaranteeing backward compatibility with an earlier release. This is helpful if it becomes necessary to revert to the earlier release.

This parameter specifies the release with which Oracle must maintain compatibility. It enables you to take advantage of the maintenance improvements of a new release immediately in your production systems without testing the new functionality in your environment. Some features of the release may be restricted.

When using a standby database, this parameter must have the same value on both the primary and standby databases.

**See Also:**

*Oracle Database Upgrade Guide* for more information on setting this parameter

# CONNECTION_BROKERS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `CONNECTION_BROKERS = broker_description[,...]`<br>**`broker_description::=`**<br>`((type_clause)(brokers_clause)[(broker_options)])`<br>**`type_clause::=`**<br>`TYPE={ DEDICATED \| EMON \| POOLED }`<br>Indicates the broker type. This is a mandatory parameter.<br>**`brokers_clause::=`**<br>`BROKERS=`*`integer`*<br>Indicates the number of brokers of the specified type. This is a mandatory parameter. The specified integer value must be greater than or equal to 0. If set to 0, brokers will not accept new connections, but existing connections will continue as normal.<br>**`broker_options::=`**<br>`CONNECTIONS=`*`integer`*<br>The maximum number of connections per broker. This is an optional parameter. If the specified integer value is greater than the maximum allowed for the platform, the maximum allowed for the platform is used. |
| **Default value** | The default value for the initialization parameter will contain the `DEDICATED` and `EMON` broker types. Certain types of brokers are only |

| Property | Description |
|---|---|
| | started when certain features are enabled, so their configuration will only apply if the broker is needed. |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |
| **Oracle RAC** | The same value should be set on all instances |

CONNECTION_BROKERS is used to specify connection broker types, the number of connection brokers of each type, and the maximum number of connections per broker.

The connection brokers are used in these cases:

- DEDICATED brokers are used by the multiprocess and multithreaded Oracle features. See *Oracle Database Concepts* for more information about these features.

  DEDICATED brokers are also used when the USE_DEDICATED_BROKER initialization parameter is set to true. See "USE_DEDICATED_BROKER" for more information about the USE_DEDICATED_BROKER initialization parameter.

- EMON brokers are used when there is a client registered for notification over 12*c* AQ queues. It is not on by default. See *Oracle Database Advanced Queuing User's Guide* for more information about AQ queues.

- POOLED brokers are used whenever a Database Resident Connection Pooling (DRCP) pool is active. See *Oracle Database Concepts* for more information about DRCP pooling.

  The POOLED broker does not appear in the parameter by default. In this case, the DRCP fields are used. If you specify the POOLED broker in the parameter, then that specification will override the DRCP fields, and further attempts to set the DRCP fields will then throw an error. Oracle recommends that the parameter be used instead of the DRCP fields.

To make it easier to modify the list, individual entries are indexed using the TYPE so it is not necessary to re-specify the entire list. For example, to adjust the number of DEDICATED brokers:

```
alter system set connection_brokers = '((TYPE=DEDICATED)
(BROKERS=2))'
```

# CONTROL_FILE_RECORD_KEEP_TIME

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 (days) |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 to 365 (days) |
| **Basic** | No |

CONTROL_FILE_RECORD_KEEP_TIME specifies the minimum number of days before a reusable record in the control file can be reused. In the event a new record must be

added to a reusable section and the oldest record has not aged enough, the record section expands. If this parameter is set to 0, then reusable sections never expand, and records are reused as needed.

If the number of reusable records in the control file exceeds the circular reuse record limit UB4MAXVAL, then reusable records will be overwritten even if CONTROL_FILE_RECORD_KEEP_TIME has not elapsed. UB4MAXVAL is defined in the oratypes.h header file, which is found in the public directory. Its value may vary according to the operating system you are using.

**Note:**

This parameter applies only to records in the control file that are circularly reusable (such as archive log records and various backup records). It does not apply to records such as data file, tablespace, and redo thread records, which are never reused unless the corresponding object is dropped from the tablespace.

**See Also:**

- "CONTROL_FILES"
- *Oracle Database Backup and Recovery User's Guide* for more information about this parameter

# CONTROL_FILES

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | CONTROL_FILES = *filename* [, *filename* ] ...<br>**Note:** The control file name can be an OMF (Oracle Managed Files) name. This occurs when the control file is re-created using the CREATE CONTROLFILE REUSE statement. |
| **Default value** | Operating system-dependent |
| **Modifiable** | No |
| **Range of values** | 1 to 8 filenames |
| **Basic** | Yes |
| **Oracle RAC** | Multiple instances must have the same value. |

Every database has a **control file**, which contains entries that describe the structure of the database (such as its name, the timestamp of its creation, and the names and locations of its data files and redo files). CONTROL_FILES specifies one or more names of control files, separated by commas.

Oracle recommends that you multiplex multiple control files on different devices or mirror the file at the operating system level.

**See Also:**

- *Oracle Database Administrator's Guide* for information about specifying control files at database creation

# CONTROL_MANAGEMENT_PACK_ACCESS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `CONTROL_MANAGEMENT_PACK_ACCESS = { NONE \| DIAGNOSTIC \| DIAGNOSTIC+TUNING }` |
| **Default value** | Enterprise Edition: `DIAGNOSTIC+TUNING`<br>All other editions: `NONE` |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`CONTROL_MANAGEMENT_PACK_ACCESS` specifies which of the Server Manageability Packs should be active. The following packs are available:

- The `DIAGNOSTIC` pack includes AWR, ADDM, and so on.
- The `TUNING` pack includes SQL Tuning Advisor, SQLAccess Advisor, and so on.

  A license for `DIAGNOSTIC` is required for enabling the `TUNING` pack.

### Values

- `NONE`

  Both packs are not available.

- `DIAGNOSTIC`

  Only the `DIAGNOSTIC` pack is available.

- `DIAGNOSTIC+TUNING`

  Both packs are available. This is the default.

**See Also:**

*Oracle Database Licensing Information*

# CORE_DUMP_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `CORE_DUMP_DEST = directory` |

| Property | Description |
|---|---|
| **Default value** | *ORACLE_HOME*/DBS |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

CORE_DUMP_DEST is primarily a UNIX parameter and may not be supported on your platform. It specifies the directory where Oracle dumps core files.

**Note:**

This parameter is ignored by the new diagnosability infrastructure introduced in Oracle Database 11g Release 1 (11.1), which places trace and core files in a location controlled by the DIAGNOSTIC_DEST initialization parameter.

**See Also:**

"DIAGNOSTIC_DEST"

# CPU_COUNT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 to unlimited |
| **Basic** | No |

CPU_COUNT specifies the number of CPUs available for Oracle Database to use. On CPUs with multiple CPU threads, it specifies the total number of available CPU threads. Various components of Oracle Database are configured based on the number of CPUs, such as the Optimizer, Parallel Query, and Resource Manager.

If CPU_COUNT is set to 0 (its default setting), then Oracle Database continuously monitors the number of CPUs reported by the operating system and uses the current count. If CPU_COUNT is set to a value other than 0, then Oracle Database will use this count rather than the actual number of CPUs, thus disabling dynamic CPU reconfiguration. When Resource Manager is enabled, setting CPU_COUNT limits the CPU utilization to approximately CPU_COUNT processors.

**Note:**

Setting CPU_COUNT to a value greater than the current number of CPUs results in an error. However, if CPU_COUNT is set to a value greater than the current number of CPUs in the initialization parameter file, then CPU_COUNT is capped to the current number of CPUs.

**See Also:**

- *Oracle Database VLDB and Partitioning Guide* for information about how CPU_COUNT is used to determine the default degree of parallelism for a single instance or Oracle RAC configuration when the PARALLEL clause is specified but no degree of parallelism is listed
- *Oracle Database Administrator's Guide* for an example of how CPU resources are allocated if you enable instance caging and set a maximum utilization limit in a resource plan

# CREATE_BITMAP_AREA_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | No |
| **Range of values** | Operating system-dependent |
| **Basic** | No |

**Note:**

Oracle does not recommend using the CREATE_BITMAP_AREA_SIZE parameter unless the instance is configured with the shared server option. Oracle recommends that you enable automatic sizing of SQL working areas by setting PGA_AGGREGATE_TARGET instead. CREATE_BITMAP_AREA_SIZE is retained for backward compatibility.

CREATE_BITMAP_AREA_SIZE is relevant only for systems containing bitmap indexes. It specifies the amount of memory (in bytes) allocated for bitmap creation. A larger value may speed up index creation.

**Cardinality** is the number of unique values in a column in relation to the number of rows in the table. If cardinality is very small, you can set a small value for this parameter. For example, if cardinality is only 2, then the value can be in kilobytes rather than megabytes. As a general rule, the higher the cardinality, the more memory is needed for optimal performance.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information on using bitmap indexes

# CREATE_STORED_OUTLINES

| Property | Description |
|---|---|
| **Parameter type** | String |

| Property | Description |
|---|---|
| **Syntax** | `CREATE_STORED_OUTLINES = { true \| false \| category_name } [NOOVERRIDE]` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |

`CREATE_STORED_OUTLINES` determines whether Oracle automatically creates and stores an outline for each query submitted during the session.

### Values

- `true`

  Enables automatic outline creation for subsequent queries in the same session. These outlines receive a unique system-generated name and are stored in the `DEFAULT` category. If a particular query already has an outline defined for it in the `DEFAULT` category, then that outline will remain and a new outline will not be created.

- `false`

  Disables automatic outline creation during the session. This is the default.

- *`category_name`*

  Enables the same behavior as `true` except that any outline created during the session is stored in the *`category_name`* category.

- `NOOVERRIDE`

  `NOOVERRIDE` specifies that this system setting will not override the setting for any session in which this parameter was explicitly set. If you do not specify `NOOVERRIDE`, then this setting takes effect in all sessions.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information on setting this parameter

## CURSOR_BIND_CAPTURE_DESTINATION

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `CURSOR_BIND_CAPTURE_DESTINATION = { off \| memory \| memory+disk }` |
| **Default value** | `NULL` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |

CURSOR_BIND_CAPTURE_DESTINATION determines the location at which bind variables that are captured from SQL cursors are available.

### Values

- off

  Bind variables are not captured from SQL cursors.

- memory

  Bind variables are captured from SQL cursors, and are available only in memory (V$ views).

- memory+disk

  Bind variables are captured from SQL cursors, and are available in memory (V$ views) and disk (Automatic Workload Repository tables, SQL Tuning Set tables, and so on).

  When you specify this value with a SQL ALTER SESSION or ALTER SYSTEM statement, enclose the value in single quotes or double quotes. For example:

```
ALTER SESSION SET CURSOR_BIND_CAPTURE_DESTINATION = 'memory+disk';
ALTER SYSTEM SET CURSOR_BIND_CAPTURE_DESTINATION = "memory+disk";
```

## CURSOR_SHARING

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | CURSOR_SHARING = { EXACT \| FORCE } |
| **Default value** | NULL |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

CURSOR_SHARING determines what kind of SQL statements can share the same cursors.

### Values

- FORCE

  Allows the creation of a new cursor if sharing an existing cursor, or if the cursor plan is not optimal.

- EXACT

  Only allows statements with identical text to share the same cursor.

---

**See Also:**

*Oracle Database SQL Tuning Guide* before setting the CURSOR_SHARING parameter to learn about the performance implications

---

# CURSOR_SPACE_FOR_TIME

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

**Note:**

The `CURSOR_SPACE_FOR_TIME` parameter is deprecated. It is retained for backward compatibility only.

`CURSOR_SPACE_FOR_TIME` lets you use more space for cursors to save time. It affects both the shared SQL area and the client's private SQL area.

Most users will not need to set this parameter because of the significantly enhanced concurrency modifications introduced in Oracle Database 10*g* Release 2 (10.2.0.2) and later.

### Values

- `true`

  Shared SQL areas are kept pinned in the shared pool. As a result, shared SQL areas are not aged out of the pool as long as an open cursor references them. Because each active cursor's SQL area is present in memory, execution is faster. However, the shared SQL areas never leave memory while they are in use. Therefore, you should set this parameter to `true` only when the shared pool is large enough to hold all open cursors simultaneously.

  In addition, a setting of `true` retains the private SQL area allocated for each cursor between executions instead of discarding it after cursor execution, saving cursor allocation and initialization time.

- `false`

  Shared SQL areas can be deallocated from the library cache to make room for new SQL statements.

**Note:**

If this parameter is set to `true`, then the `SERIAL_REUSE` parameter is disabled.

**See Also:**

- "SERIAL_REUSE"

# DB_nK_CACHE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `DB_[2 \| 4 \| 8 \| 16 \| 32]K_CACHE_SIZE = integer [K \| M \| G]` |
| **Default value** | `0` (additional block size caches are not configured by default) |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Minimum: `0` (values greater than zero are automatically modified to be either the granule size * number of processor groups, or 4 MB * number of CPUs, whichever is greater)<br>Maximum: operating system-dependent |
| **Basic** | No |

`DB_nK_CACHE_SIZE` (where *n* = 2, 4, 8, 16, 32) specifies the size of the cache for the *n*K buffers. You can set this parameter only when `DB_BLOCK_SIZE` has a value other than *n*K. For example, if `DB_BLOCK_SIZE=4096`, then it is illegal to specify the parameter `DB_4K_CACHE_SIZE` (because the size for the 4 KB block cache is already specified by `DB_CACHE_SIZE`).

Do not set this parameter to zero if there are any online tablespaces with an *n*K block size.

Operating system-specific block size restrictions apply. For example, you cannot set `DB_32K_CACHE_SIZE` if the operating system's maximum block size is less than 32 KB. Also, you cannot set `DB_2K_CACHE_SIZE` if the minimum block size is greater than 2 KB.

**See Also:**

Your operating system-specific Oracle documentation for more information on block size restrictions

# DB_BIG_TABLE_CACHE_PERCENT_TARGET

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_BIG_TABLE_CACHE_PERCENT_TARGET = string` |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |

| Property | Description |
|---|---|
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values, but it is recommended to keep the big table cache section size uniform. |

`DB_BIG_TABLE_CACHE_PERCENT_TARGET` specifies the cache section target size for automatic big table caching, as a percentage of the buffer cache. Automatic big table caching enables parallel queries and serial queries to use the buffer cache, which enhances the in-memory query capabilities of Oracle Database. Automatic big table caching is designed primarily to enhance performance for data warehouse workloads, but it also improves performance in mixed workloads.

Starting in Oracle Database 12*c* Release 1 (12.1.0.2), table scans can use a different algorithm in the following scenarios:

- Parallel queries:

  In single-instance and Oracle Real Application Clusters (Oracle RAC) databases, parallel queries can use the automatic big table cache when the `DB_BIG_TABLE_CACHE_PERCENT_TARGET` initialization parameter is set to a non-zero value, and the `PARALLEL_DEGREE_POLICY` initialization parameter is set to `AUTO` or `ADAPTIVE`.

- Serial queries:

  In a single-instance configuration only, serial queries can use the automatic big table cache when the `DB_BIG_TABLE_CACHE_PERCENT_TARGET` initialization parameter is set to a non-zero value.

When a non-zero value is specified for the `DB_BIG_TABLE_CACHE_PERCENT_TARGET` parameter, the value indicates the percentage of the buffer cache to reserve for the big table cache. The largest value that can be specified with the `DB_BIG_TABLE_CACHE_PERCENT_TARGET` parameter is 90, which reserves 10% of the buffer cache for usage besides table scans.

The default value of this parameter is `0`. Therefore, automatic big table caching is not enabled by default. When automatic big table caching is not enabled, queries might run using the least recently used (LRU) mechanism for cached reads, or might decide to use direct reads for the table scan.

If a large table is about the size of the combined size of the big table cache of all instances, the table will be partitioned and cached or mostly cached on all instances. With in-memory parallel query, this could eliminate most disk reads for queries on the table, or the database could intelligently read from disk only for the portion of the table that does not fit in the big table cache. If the big table cache cannot cache all the tables to be scanned, only the most frequently accessed tables will be cached, and the rest will be read via direct read automatically.

Use these guidelines when setting the parameter:

- If you do not enable automatic degree of parallelism (DOP) in your Oracle RAC environment, do not set this parameter because the big table cache is not used in that situation.

- When setting this parameter, consider the workload mix: how much of the workload is for OLTP; insert, update, and random access; and how much of the workload involves table scans. Because data warehouse workloads often perform

large table scans, you may consider giving the big table cache section a higher percentage of buffer cache space for data warehouses.

- This parameter can be dynamically changed if the workload changes. The change could take some time to reach the target (depending on the current workload) because buffer cache memory might be actively used at that time.

**Note:**

Automatic big table caching uses temperature and object-based algorithms to track medium and big tables. Oracle will cache very small tables, but they will not be tracked by automatic big table caching.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "PARALLEL_DEGREE_POLICY"
- See "V$BT_SCAN_CACHE" and "V$BT_SCAN_OBJ_TEMPS" for more information about the big table cache
- *Oracle Database VLDB and Partitioning Guide* for more information about this parameter and about automatic big table caching

# DB_BLOCK_BUFFERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | No |
| **Range of values** | `50` to an operating system-specific maximum |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values, and you can change the values as needed. |

**Note:**

`DB_BLOCK_BUFFERS` cannot be combined with the dynamic `DB_CACHE_SIZE` parameter; combining these parameters in the same parameter file will produce an error.

DB_BLOCK_BUFFERS specifies the number of database buffers in the buffer cache. It is one of several parameters that contribute to the total memory requirements of the SGA of an instance.

This parameter, together with DB_BLOCK_SIZE, determines the total size of the buffer cache. Effective use of the buffer cache can greatly reduce the I/O load on the database. DB_BLOCK_SIZE can be specified only when the database is first created, so you use DB_BLOCK_BUFFERS to control the size of the buffer cache.

# DB_BLOCK_CHECKING

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | DB_BLOCK_CHECKING = { FALSE \| OFF \| LOW \| MEDIUM \| TRUE \| FULL } |
| **Default value** | FALSE |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

DB_BLOCK_CHECKING specifies whether Oracle Database performs block checking for database blocks.

### Values

- OFF or FALSE

  No block checking is performed for blocks in user tablespaces. However, semantic block checking for SYSTEM tablespace blocks is always turned on.

- LOW

  Basic block header checks are performed after block contents change in memory (for example, after UPDATE, INSERT or DELETE statements, or after inter-instance block transfers in Oracle RAC).

- MEDIUM

  All LOW checks and full semantic checks are performed for all objects except indexes (whose contents can be reconstructed by a drop+rebuild on encountering a corruption).

- FULL or TRUE

  All LOW and MEDIUM checks and full semantic checks are performed for all objects.

Oracle checks a block by going through the data in the block, making sure it is logically self-consistent. Block checking can often prevent memory and data corruption. Block checking typically causes 1% to 10% overhead in most applications, depending on workload and the parameter value. Specific DML overhead may be higher. The more updates or inserts in a workload, the more expensive it is to turn on block checking. You should set DB_BLOCK_CHECKING to FULL if the performance overhead is acceptable.

For backward compatibility, the use of FALSE (implying OFF) and TRUE (implying FULL) is preserved.

**See Also:**

*Oracle Database Administrator's Guide* for more information about this parameter

# DB_BLOCK_CHECKSUM

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_BLOCK_CHECKSUM = { OFF \| FALSE \| TYPICAL \| TRUE \| FULL }` |
| **Default value** | `TYPICAL` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |

`DB_BLOCK_CHECKSUM` determines whether `DBW`*n* and the direct loader will calculate a checksum (a number calculated from all the bytes stored in the block) and store it in the cache header of every data block when writing it to disk. Checksums are verified when a block is read - only if this parameter is `TYPICAL` or `FULL` and the last write of the block stored a checksum. In `FULL` mode, Oracle also verifies the checksum before a change application from update/delete statements and recomputes it after the change is applied. In addition, Oracle gives every log block a checksum before writing it to the current log.

Most of the log block checksum is done by the generating foreground processes, while the LGWR or the LGWR slave processes (LG*nn* processes) perform the rest of the work, for better CPU and cache efficiency.

If this parameter is set to `OFF`, `DBW`*n* calculates checksums only for the `SYSTEM` tablespace, but not for user tablespaces. In addition, no log checksum is performed when this parameter is set to `OFF`.

Checksums allow Oracle to detect corruption caused by underlying disks, storage systems, or I/O systems. If set to `FULL`, `DB_BLOCK_CHECKSUM` also catches in-memory corruptions and stops them from making it to the disk. Turning on this feature in `TYPICAL` mode causes only an additional 1% to 2% overhead. In the `FULL` mode it causes 4% to 5% overhead. Oracle recommends that you set `DB_BLOCK_CHECKSUM` to `TYPICAL`.

For backward compatibility the use of `TRUE` (implying `TYPICAL`) and `FALSE` (implying `OFF`) values is preserved.

**See Also:**

*Oracle Database Backup and Recovery Reference* for more information about this parameter

# DB_BLOCK_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 8192 |
| **Modifiable** | No |
| **Range of values** | 2048 to 32768, but your operating system may have a narrower range |
| **Basic** | Yes |
| **Oracle RAC** | You must set this parameter for every instance, and multiple instances must have the same value. |

**Note:**

Set this parameter at the time of database creation. Do not alter it afterward.

DB_BLOCK_SIZE specifies (in bytes) the size of Oracle database blocks. Typical values are 4096 and 8192. The value of this parameter must be a multiple of the physical block size at the device level.

The value for DB_BLOCK_SIZE in effect at the time you create the database determines the size of the blocks. The value must remain set to its initial value.

For Oracle Real Application Clusters (Oracle RAC), this parameter affects the maximum value of the FREELISTS storage parameter for tables and indexes. Oracle uses one database block for each freelist group. Decision support system (DSS) and data warehouse database environments tend to benefit from larger block size values.

**Note:**

64-bit operating systems support a maximum DB_BLOCK_SIZE value of 32768

**See Also:**

- *Oracle Database Administrator's Guide* for information on setting this parameter
- *Oracle Database SQL Language Reference* for information on freelist groups

# DB_CACHE_ADVICE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | DB_CACHE_ADVICE = { ON \| READY \| OFF } |

| Property | Description |
|---|---|
| **Default value** | If STATISTICS_LEVEL is set to TYPICAL or ALL, then ON<br>If STATISTICS_LEVEL is set to BASIC, then OFF |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

DB_CACHE_ADVICE enables or disables statistics gathering used for predicting behavior with different cache sizes through the V$DB_CACHE_ADVICE performance view.

### Values

- OFF

  Advisory is turned off and the memory for the advisory is not allocated.

- READY

  Advisory is turned off but the memory for the advisory remains allocated. Allocating the memory before the advisory is actually turned on avoids the risk of an error when you switch the parameter to ON.

  If the parameter is switched to this state from ON, the contents of the view are preserved and the memory for the advisory is retained.

  If the parameter is switched to this state from OFF, you may get an error.

- ON

  Advisory is turned on. CPU and memory overheads are incurred. Attempting to set the parameter to this state when it is already in the OFF state may result in an error. Otherwise, the view (V$DB_CACHE_ADVICE) is reset and statistics are gathered to the newly refreshed view.

  If the parameter is in the READY state, you can set it to ON without any errors because the memory is already allocated. The view is reset and statistics are displayed in the newly refreshed view.

**See Also:**

"V$DB_CACHE_ADVICE"

## DB_CACHE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | DB_CACHE_SIZE = *integer* [K \| M \| G] |
| **Default value** | If SGA_TARGET is set: If the parameter is not specified, then the default is 0 (internally determined by the Oracle Database). If the parameter is specified, then the user-specified value indicates a minimum value for the memory pool. |

| Property | Description |
|---|---|
| | If SGA_TARGET is not set, then the default is either 48 MB or 4 MB * number of CPUs, whichever is greater |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

DB_CACHE_SIZE specifies the size of the DEFAULT buffer pool for buffers with the primary block size (the block size defined by the DB_BLOCK_SIZE initialization parameter).

The value must be at least 4M * number of cpus (smaller values are automatically rounded up to this value). A user-specified value larger than this is rounded up to the nearest granule size. A value of zero is illegal because it is needed for the DEFAULT memory pool of the primary block size, which is the block size for the SYSTEM tablespace.

**See Also:**

*Oracle Database Performance Tuning Guide* and *Oracle Database Administrator's Guide* for more information on setting this parameter

# DB_CREATE_FILE_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | DB_CREATE_FILE_DEST = *directory* \| *disk group* |
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | Yes |

DB_CREATE_FILE_DEST specifies the default location for Oracle-managed datafiles. This location is also used as the default location for Oracle-managed control files and online redo logs if none of the DB_CREATE_ONLINE_LOG_DEST_*n* initialization parameters are specified.

If a file system directory is specified as the default location, then the directory must already exist; Oracle does not create it. The directory must have appropriate permissions that allow Oracle to create files in it. Oracle generates unique names for the files, and a file thus created is an Oracle Managed File.

**See Also:**

*Oracle Database Administrator's Guide* for more information on setting this parameter and on Oracle Managed Files

# DB_CREATE_ONLINE_LOG_DEST_*n*

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_CREATE_ONLINE_LOG_DEST_[1 \| 2 \| 3 \| 4 \| 5] = directory \| disk group` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | Yes |

`DB_CREATE_ONLINE_LOG_DEST_n` (where *n* = 1, 2, 3, ... 5) specifies the default location for Oracle-managed control files and online redo logs. If more than one `DB_CREATE_ONLINE_LOG_DEST_n` parameter is specified, then the control file or online redo log is multiplexed across the locations of the other `DB_CREATE_ONLINE_LOG_DEST_n` parameters. One member of each online redo log is created in each location, and one control file is created in each location.

Specifying at least two parameters provides greater fault tolerance for the control files and online redo logs if one of the locations should fail.

If a file system directory is specified as the default location, then the directory must already exist; Oracle does not create it. The directory must have appropriate permissions that allow Oracle to create files in it. Oracle generates unique names for the files, and a file thus created is an Oracle Managed File.

**See Also:**

*Oracle Database Administrator's Guide* for more information on setting this parameter and on Oracle Managed Files

# DB_DOMAIN

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_DOMAIN = domain_name` |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | Any legal string of name components, separated by periods and up to 128 characters long (including the periods). |
| **Basic** | Yes |
| **Oracle RAC** | You must set this parameter for every instance, and multiple instances must have the same value. |

In a distributed database system, `DB_DOMAIN` specifies the logical location of the database within the network structure. You should set this parameter if this database

is or ever will be part of a distributed system. The value consists of the extension components of a global database name, consisting of valid identifiers (any alphanumeric ASCII characters), separated by periods.

**Note:**

Oracle recommends that you specify DB_DOMAIN as a unique string for all databases in a domain.

This parameter allows one department to create a database without worrying that it might have the same name as a database created by another department. If one sales department's DB_DOMAIN is JAPAN.ACME.COM, then their SALES database (SALES.JAPAN.ACME.COM) is uniquely distinguished from another database with DB_NAME = SALES but with DB_DOMAIN = US.ACME.COM.

If you omit the domains from the name of a database link, Oracle expands the name by qualifying the database with the domain of your local database as it currently exists in the data dictionary, and then stores the link name in the data dictionary. The characters valid in a database domain name are alphanumeric characters and the number sign (#) character.

**See Also:**

- *Oracle Database Administrator's Guide* for more information on setting this parameter
- The data dictionary view "GLOBAL_NAME"

# DB_FILE_MULTIBLOCK_READ_COUNT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | The default value corresponds to the maximum I/O size that can be efficiently performed and is platform-dependent |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | Operating system-dependent |
| **Basic** | No |

DB_FILE_MULTIBLOCK_READ_COUNT is one of the parameters you can use to minimize I/O during table scans. It specifies the maximum number of blocks read in one I/O operation during a sequential scan. The total number of I/Os needed to perform a full table scan depends on such factors as the size of the table, the multiblock read count, and whether parallel execution is being utilized for the operation.

The default value of this parameter is a value that corresponds to the maximum I/O size that can be performed efficiently. This value is platform-dependent and is 1MB for most platforms.

Because the parameter is expressed in blocks, it will be set to a value that is equal to the maximum I/O size that can be performed efficiently divided by the standard block size. Note that if the number of sessions is extremely large the multiblock read count value is decreased to avoid the buffer cache getting flooded with too many table scan buffers.

Even though the default value may be a large value, the optimizer will not favor large I/Os if you do not set this parameter. It will favor large I/Os only if you explicitly set this parameter to a large value.

Online transaction processing (OLTP) and batch environments typically have values in the range of 4 to 16 for this parameter. DSS and data warehouse environments tend to benefit most from maximizing the value of this parameter. The optimizer is more likely to choose a full table scan over an index if the value of this parameter is high.

The maximum value is the operating system's maximum I/O size expressed as Oracle blocks ((max I/O size)/`DB_BLOCK_SIZE`). If you set this parameter to a value greater than the maximum, then Oracle uses the maximum.

**See Also:**

*Oracle Database Performance Tuning Guide* for information about how setting this parameter incorrectly can cause excessive I/O waits for some execution plans

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information on setting this parameter

# DB_FILE_NAME_CONVERT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_FILE_NAME_CONVERT = '`*`string1`*`' , '`*`string2`*`' , '`*`string3`*`' , '`*`string4`*`' , ...`<br>Where:<br>• *`string1`* is the pattern of the primary database filename<br>• *`string2`* is the pattern of the standby database filename<br>• *`string3`* is the pattern of the primary database filename<br>• *`string4`* is the pattern of the standby database filename<br>You can use as many pairs of primary and standby replacement strings as required. You can use single or double quotation marks.<br>The following are example settings that are acceptable:<br>`DB_FILE_NAME_CONVERT = '/dbs/t1/','/dbs/t1/ s','dbs/t2/ ','dbs/t2/s_'` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SESSION` |
| **Basic** | No |

DB_FILE_NAME_CONVERT is useful for creating a duplicate database for recovery purposes. It converts the filename of a new datafile on the primary database to a filename on the standby database. If you add a datafile to the primary database, you must add a corresponding file to the standby database. When the standby database is updated, this parameter converts the datafile name on the primary database to the datafile name on the standby database. The file on the standby database must exist and be writable, or the recovery process will halt with an error.

If you specify an odd number of strings (the last string has no corresponding replacement string), an error is signalled during startup. If the filename being converted matches more than one pattern in the pattern/replace string list, the first matched pattern takes effect. There is no limit on the number of pairs that you can specify in this parameter (other than the hard limit of the maximum length of multivalue parameters).

Set the value of this parameter to two strings. The first string is the pattern found in the datafile names on the primary database. The second string is the pattern found in the datafile names on the standby database.

You can also use DB_FILE_NAME_CONVERT to rename the datafiles in the clone control file when setting up a clone database during tablespace point-in-time recovery.

**See Also:**

- *Oracle Database Backup and Recovery User's Guide* for more information about database backup and recovery
- *Oracle Data Guard Concepts and Administration* for more information about Oracle Data Guard

# DB_FILES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 200 |
| **Modifiable** | No |
| **Range of values** | Minimum: the largest among the absolute file numbers of the datafiles in the database<br>Maximum: operating system-dependent |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must have the same value. |

DB_FILES specifies the maximum number of database files that can be opened for this database. The maximum valid value is the maximum number of files, subject to operating system constraint, that will ever be specified for the database, including files to be added by ADD DATAFILE statements.

If you increase the value of DB_FILES, then you must shut down and restart all instances accessing the database before the new value can take effect. If you have a primary and standby database, then they should have the same value for this parameter.

**See Also:**

- *Oracle Real Application Clusters Administration and Deployment Guide* for information on setting this parameter in an Oracle RAC environment
- Your operating system-specific Oracle documentation for the default value of this parameter

# DB_FLASH_CACHE_FILE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_FLASH_CACHE_FILE = filename [,filename]... \| disk group` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`DB_FLASH_CACHE_FILE` specifies file name(s) for the flash memory or disk group representing a collection of flash memory, for use with Database Smart Flash Cache.

You can specify up to 16 file names for flash memory devices. For example, if there are three flash raw devices:

```
db_flash_cache_file = /dev/raw/sda, /dev/raw/sdb, /dev/raw/sdc
```

Specifying this parameter without also specifying the `DB_FLASH_CACHE_SIZE` initialization parameter is not allowed.

**See Also:**

"DB_FLASH_CACHE_SIZE"

# DB_FLASH_CACHE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `DB_FLASH_CACHE_SIZE = integer [K \| M \| G] [, integer [K \| M \| G]]...` |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Minimum: `0`<br>Maximum: operating system-dependent |
| **Basic** | No |

DB_FLASH_CACHE_SIZE specifies the size of the Database Smart Flash Cache (flash cache). This parameter may only be specified at instance startup.

You can specify up to 16 file sizes, for each of the flash memory devices specified with DB_FLASH_CACHE_FILE. For example, if there are three flash raw devices, you can specify the sizes of each device as follows:

```
db_flash_cache_file = /dev/raw/sda, /dev/raw/sdb, /dev/raw/sdc
db_flash_cache_size = 32G, 32G, 64G
```

If your flash cache consists of one flash cache device, you can dynamically change this parameter to 0 for that flash cache device (disabling the flash cache) after the database is started. You can then reenable the flash cache by setting this parameter for the device back to the original value when the database was started. Dynamic resizing of DB_FLASH_CACHE_SIZE or reenabling flash cache to a different size is not supported.

If your flash cache includes multiple flash cache devices, you can dynamically change the parameter to 0 for a particular flash cache device (turning it off) after the database is started. You can then reenable that flash cache device by setting this parameter for the device back to the original value it had when the database was started (turning it back on).

For example, to turn off the /dev/raw/sdb flash cache device:

```
db_flash_cache_file = /dev/raw/sda, /dev/raw/sdb, /dev/raw/sdc
db_flash_cache_size = 32G, 0, 64G
```

And, to turn the /dev/raw/sdb flash cache device back on again:

```
db_flash_cache_file = /dev/raw/sda, /dev/raw/sdb, /dev/raw/sdc
db_flash_cache_size = 32G, 32G, 64G
```

**See Also:**

"DB_FLASH_CACHE_FILE"

# DB_FLASHBACK_RETENTION_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 131072 (minutes) |
| **Modifiable** | ALTER SYSTEM ... SID='*' |
| **Range of values** | 0 to $2^{31}$ - 1 |
| **Basic** | No |

DB_FLASHBACK_RETENTION_TARGET specifies the upper limit (in minutes) on how far back in time the database may be flashed back. How far back one can flashback a database depends on how much flashback data Oracle has kept in the fast recovery area.

**See Also:**

*Oracle Data Guard Broker* for enabling Flashback Database on Oracle Data Guard primary and target standby databases

# DB_INDEX_COMPRESSION_INHERITANCE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_INDEX_COMPRESSION_INHERITANCE = { TABLESPACE \| TABLE \| ALL \| NONE }` |
| **Default value** | `NONE` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | You must set this parameter for every instance, and multiple instances must have the same value. |

`DB_INDEX_COMPRESSION_INHERITANCE` displays the options that are set for table or tablespace level compression inheritance.

**Values:**

- `TABLESPACE`

  Indexes will be created with ADVANCED LOW compression if the tablespace they reside in has OLTP compression enabled as the default.

- `TABLE`

  If the underlying table is OLTP compressed, the indexes on this table will be ADVANCED LOW compressed.

- `ALL`

  If either the underlying table or tablespace is OLTP compressed, the indexes on this table will be ADVANCED LOW compressed.

- `NONE`

  Indexes will not inherit any compression from the base table or its tablespace. They will only be compressed if users specify it.

**See Also:**

*Oracle Database Administrator's Guide* for more information about index compression

# DB_KEEP_CACHE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `DB_KEEP_CACHE_SIZE = integer [K \| M \| G]` |
| **Default value** | `0` (`DB_KEEP_CACHE_SIZE` is not configured by default) |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Minimum: `0` (values greater than zero are automatically modified to be either the granule size * number of processor groups, or 4 MB * number of CPUs, whichever is greater)<br>Maximum: operating system-dependent |
| **Basic** | No |

`DB_KEEP_CACHE_SIZE` specifies the size of the `KEEP` buffer pool. The size of the buffers in the `KEEP` buffer pool is the primary block size (the block size defined by the `DB_BLOCK_SIZE` initialization parameter).

When you modify the value of `DB_KEEP_CACHE_SIZE` using `ALTER SYSTEM`, you must restart the database for the new value to take effect.

**See Also:**

- "DB_RECYCLE_CACHE_SIZE"
- *Oracle Database Performance Tuning Guide* for information on setting these parameters and on using multiple buffer pools

# DB_LOST_WRITE_PROTECT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_LOST_WRITE_PROTECT = { NONE \| TYPICAL \| FULL }` |
| **Default value** | `NONE` |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | In Oracle RAC instances, the parameter value is systemwide. |

`DB_LOST_WRITE_PROTECT` enables or disables lost write detection. A data block lost write occurs when an I/O subsystem acknowledges the completion of the block write, while in fact the write did not occur in the persistent storage.

When the parameter is set to `TYPICAL` on the primary database, the instance logs buffer cache reads for read/write tablespaces in the redo log, which is necessary for detection of lost writes.

When the parameter is set to `FULL` on the primary database, the instance logs reads for read-only tablespaces and read/write tablespaces.

When the parameter is set to TYPICAL or FULL on the standby database or on the primary database during media recovery, the instance performs lost write detection.

When the parameter is set to NONE on either the primary database or the standby database, no lost write detection functionality is enabled.

# DB_NAME

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | DB_NAME = *database_name* |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Basic** | Yes |
| **Oracle RAC** | You must set this parameter for every instance. Multiple instances must have the same value, or the same value must be specified in the STARTUP OPEN SQL*Plus command or the ALTER DATABASE MOUNT SQL statement. |

DB_NAME specifies a database identifier of up to 8 characters. This parameter must be specified and must correspond to the name specified in the CREATE DATABASE statement.

If you have multiple databases, the value of this parameter should match the Oracle instance identifier of each one to avoid confusion with other databases running on the system. The value of DB_NAME should be the same in both the standby and production initialization parameter files.

The database name specified in either the STARTUP command or the ALTER DATABASE ... MOUNT statement for each instance of the cluster database must correspond to the DB_NAME initialization parameter setting.

The following characters are valid in a database name: alphanumeric characters, underscore (_), number sign (#), and dollar sign ($). No other characters are valid. The database name must start with an alphabetic character. Oracle removes double quotation marks before processing the database name. Therefore you cannot use double quotation marks to embed other characters in the name. The database name is case insensitive.

**See Also:**

*Oracle Database Administrator's Guide* and *Oracle Real Application Clusters Administration and Deployment Guide* for more information on setting this parameter

# DB_RECOVERY_FILE_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |

| Property | Description |
|---|---|
| **Syntax** | `DB_RECOVERY_FILE_DEST = directory \| disk group` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM ... SID='*'` |
| **Basic** | Yes |
| **Oracle RAC** | You must set this parameter for every instance, and multiple instances must have the same value. |

`DB_RECOVERY_FILE_DEST` specifies the default location for the fast recovery area. The fast recovery area contains multiplexed copies of current control files and online redo logs, as well as archived redo logs, flashback logs, and RMAN backups.

Specifying this parameter without also specifying the `DB_RECOVERY_FILE_DEST_SIZE` initialization parameter is not allowed.

**See Also:**

- *Oracle Database Backup and Recovery User's Guide* for information on setting up and configuring the fast recovery area
- "DB_RECOVERY_FILE_DEST_SIZE"

# DB_RECOVERY_FILE_DEST_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `DB_RECOVERY_FILE_DEST_SIZE = integer [K \| M \| G]` |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM ... SID='*'` |
| **Basic** | Yes |
| **Oracle RAC** | You must set this parameter for every instance, and multiple instances must have the same value. |

`DB_RECOVERY_FILE_DEST_SIZE` specifies (in bytes) the hard limit on the total space to be used by target database recovery files created in the fast recovery area.

Note that neither block 0 nor the OS block header of each Oracle file is included in this size. Allow an extra 10% for this data when computing the actual disk usage required for the fast recovery area.

**See Also:**

- *Oracle Database Backup and Recovery User's Guide* for information on setting up and configuring the fast recovery area
- "DB_RECOVERY_FILE_DEST"

# DB_RECYCLE_CACHE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `DB_RECYCLE_CACHE_SIZE = integer [K \| M \| G]` |
| **Default value** | `0` (`DB_RECYCLE_CACHE_SIZE` is not configured by default) |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Minimum: `0` (values greater than zero are automatically modified to be either the granule size * number of processor groups, or 4 MB * number of CPUs, whichever is greater)<br>Maximum: operating system-dependent |
| **Basic** | No |

`DB_RECYCLE_CACHE_SIZE` specifies the size of the `RECYCLE` buffer pool. The size of the buffers in the `RECYCLE` pool is the primary block size (the block size defined by the `DB_BLOCK_SIZE` initialization parameter).

**See Also:**

- "DB_KEEP_CACHE_SIZE"
- *Oracle Database Performance Tuning Guide* for information on setting these parameters and on using multiple buffer pools

# DB_SECUREFILE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_SECUREFILE = { NEVER \| PERMITTED \| PREFERRED \| ALWAYS \| IGNORE }` |
| **Default value** | `PERMITTED if the COMPATIBLE initialization parameter is set to 11.2.0.1, 11.2.0.2, or 11.2.0.3, or PREFERRED if the COMPATIBLE initialization parameter is set to 12.0.0.0 or higher` |
| **Modifiable** | `ALTER SESSION, ALTER SYSTEM` |

| Property | Description |
|---|---|
| **Basic** | No |

DB_SECUREFILE specifies whether to treat LOB files as SecureFiles.

### Values

- NEVER

  Any LOBs that are specified as SecureFiles are created as BasicFiles LOBs. All SecureFiles-specific storage options and features (for example, compress, encrypt, deduplicate) will cause an exception. The BasicFiles LOB defaults will be used for storage options not specified.

- PERMITTED

  LOBs are allowed to be created as SecureFiles.

- PREFERRED

  All LOBs are created as SecureFiles unless BASICFILE is explicitly specified in the LOB storage clause or the tablespace is a Manual Segment Space Management tablespace. When PREFERRED is set, cases where BASICFILE would otherwise be inherited from the partition or column level LOB storage are ignored; the LOBs will be created as SecureFiles instead.

- ALWAYS

  Attempts to create all LOBs as SecureFiles LOBs but creates any LOBs not in an Automatic Segment Space Managed (ASSM) tablespace as BasicFiles LOBs, unless SECUREFILE is explicitly specified. Any BasicFiles LOB storage options that are specified will be ignored and the SecureFiles LOB defaults will be used for all storage options not specified.

- IGNORE

  The SECUREFILE keyword and all SecureFiles options are ignored.

If the COMPATIBLE initialization parameter is not set to 11.1.0 or higher, then LOBs are not treated as SecureFiles.

If there is a LOB column with two partitions (one that has a tablespace for which ASSM is enabled and one that has a tablespace for which ASSM is not enabled), then LOBs in the partition with the ASSM-enabled tablespace will be treated as SecureFiles and LOBs in the other partition will be treated as BasicFiles.

If an application is not certified with SecureFiles, the value of the DB_SECUREFILE initialization parameter can be set to PERMITTED to get the default behavior of Oracle Database 11*g*.

**See Also:**

*Oracle Database SecureFiles and Large Objects Developer's Guide* for an example of setting this parameter using the ALTER SYSTEM statement

# DB_ULTRA_SAFE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_ULTRA_SAFE = { OFF \| DATA_ONLY \| DATA_AND_INDEX }` |
| **Default value** | `OFF` |
| **Modifiable** | No |
| **Basic** | No |

`DB_ULTRA_SAFE` sets the default values for other parameters that control protection levels.

### Values

- `OFF`

  When any of `DB_BLOCK_CHECKING`, `DB_BLOCK_CHECKSUM`, or `DB_LOST_WRITE_PROTECT` are explicitly set, no changes are made.

- `DATA_ONLY`
  - `DB_BLOCK_CHECKING` will be set to `MEDIUM`.
  - `DB_LOST_WRITE_PROTECT` will be set to `TYPICAL`.
  - `DB_BLOCK_CHECKSUM` will be set to `FULL`.
- `DATA_AND_INDEX`
  - `DB_BLOCK_CHECKING` will be set to `FULL`.
  - `DB_LOST_WRITE_PROTECT` will be set to `TYPICAL`.
  - `DB_BLOCK_CHECKSUM` will be set to `FULL`.

**See Also:**

- "DB_BLOCK_CHECKING"
- "DB_BLOCK_CHECKSUM"
- "DB_LOST_WRITE_PROTECT"

# DB_UNIQUE_NAME

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DB_UNIQUE_NAME = database_unique_name` |

| Property | Description |
|---|---|
| **Default value** | Database instances: the value of DB_NAME<br>Oracle Automatic Storage Management instances: +ASM |
| **Modifiable** | No |
| **Basic** | Yes |
| **Oracle RAC** | Multiple instances must have the same value. |

DB_UNIQUE_NAME specifies a globally unique name for the database. Databases with the same DB_NAME within the same DB_DOMAIN (for example, copies of a database created for reporting or a physical standby) must have a unique DB_UNIQUE_NAME. Every database's DB_UNIQUE_NAME must be unique within the enterprise.

The value of DB_UNIQUE_NAME can be up to 30 characters and is case insensitive. The following characters are valid in a database name: alphanumeric characters, underscore (_), number sign (#), and dollar sign ($).

**Note:**

DB_UNIQUE_NAME is used by several components within an Oracle instance to default file names or file paths, such as the default for the DG_BROKER_CONFIG_FILE*n* initialization parameter. In these cases, on UNIX platforms, the dollar sign ($) character will be removed from the path or file name, because the character is used by Oracle to define the start of an environment variable substitution within a path or file name.

**Note:**

As part of their operations, some database tools or utilities create a string that uniquely identifies a database. The string may include the DB_UNIQUE_NAME for a database, and other identifying information for the database, such as the database SID. Oracle Database restricts some identifiers to 30 characters, so using a short DB_UNIQUE_NAME can help prevent ORA-00972 "identifier is too long" messages from database tools and utilities that create a string that includes the DB_UNIQUE_NAME.

**See Also:**

*Oracle Data Guard Concepts and Administration* and *Oracle Database Administrator's Guide* for more information on setting this parameter

# DB_UNRECOVERABLE_SCN_TRACKING

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |

| Property | Description |
|---|---|
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must have the same value |

`DB_UNRECOVERABLE_SCN_TRACKING` enables or disables the tracking of unrecoverable (`NOLOGGING`) direct-path INSERT and direct-path load operations.

When the value is set to `true`, updates are made to the controlfile that maintains the `V$DATAFILE.UNRECOVERABLE_CHANGE#` and `V$DATAFILE.UNRECOVERABLE_TIME` columns. When the value is set to `false`, updates are not made to the controlfile. Setting this parameter to `false` may improve performance of direct-path `NOLOGGING` operations.

# DB_WRITER_PROCESSES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `1` or `CPU_COUNT / 8`, whichever is greater. If the number of processor groups is less than 100 but greater than the number of Database Writer Processes, then the number of Database Writer Processes is adjusted to be a multiple of the number of processor groups. If the number of Database Writer Processes is greater than or equal to the number of processor groups, then there is no adjustment. |
| **Modifiable** | No |
| **Range of values** | 1 to 100 |
| **Basic** | No |

`DB_WRITER_PROCESSES` specifies the initial number of Database Writer Processes for an instance. This parameter is useful for systems that modify data heavily.

There can be 1 to 100 Database Writer Processes. The names of the first 36 Database Writer Processes are DBW0-DBW9 and DBWa-DBWz. The names of the 37th through 100th Database Writer Processes are BW36-BW99.

**See Also:**

- Background Processes for additional information about the Database Writer Process
- *Oracle Database Concepts* for information on when the Database Writer Process writes dirty buffers to disk

# DBFIPS_140

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | All instances must use the same value. |

`DBFIPS_140` enables Transparent Data Encryption (TDE) and `DBMS_CRYPTO` PL/SQL package program units to run in a mode compliant to the Federal Information Processing Standard (subsequently known as "FIPS mode").

Set this parameter to `true` to use TDE and `DBMS_CRYPTO` in FIPS mode. This means that only FIPS-compliant algorithms may be used. By default, this parameter is set to `false`. When this parameter is set to `false`, all algorithms (FIPS-compliant or not) may be used.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- *Oracle Database Security Guide* for information about configuring Oracle Database FIPS 140 settings
- *Oracle Database Security Guide* for a table that describes the effect of setting the value of `DBFIPS_140` to `true` or `false` on different platforms

# DBWR_IO_SLAVES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | No |
| **Range of values** | 0 to operating system-dependent |
| **Basic** | No |

`DBWR_IO_SLAVES` is relevant only on systems with only one database writer process (`DBW0`). It specifies the number of I/O server processes used by the `DBW0` process. The `DBW0` process and its server processes always write to disk. By default, the value is `0` and I/O server processes are not used.

If you set DBWR_IO_SLAVES to a nonzero value, the number of I/O server processes used by the ARCH and LGWR processes is set to 4. However, the number of I/O server processes used by Recovery Manager is set to 4 only if asynchronous I/O is disabled (either your platform does not support asynchronous I/O or disk_asynch_io is set to false).

Typically, I/O server processes are used to simulate asynchronous I/O on platforms that do not support asynchronous I/O or that implement it inefficiently. However, you can use I/O server processes even when asynchronous I/O is being used. In that case the I/O server processes will use asynchronous I/O.

I/O server processes are also useful in database environments with very large I/O throughput, even if asynchronous I/O is enabled.

**See Also:**

- "BACKUP_TAPE_IO_SLAVES"
- *Oracle Database Performance Tuning Guide* for more information about this parameter

# DDL_LOCK_TIMEOUT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 1 |
| **Modifiable** | ALTER SESSION |
| **Range of values** | 0 to 1,000,000 (in seconds) |
| **Basic** | No |

DDL_LOCK_TIMEOUT specifies a time limit for how long DDL statements will wait in a DML lock queue. A value of zero indicates a status of NOWAIT. The maximum value of 1,000,000 seconds will result in the DDL statement waiting forever to acquire a DML lock.

If a lock is not acquired before the timeout period expires, then an error is returned.

**See Also:**

- *Oracle Database Administrator's Guide* for more information about the DDL_LOCK_TIMEOUT parameter
- *Oracle Database Development Guide* for information about nonblocking and blocking DDL

# DEFERRED_SEGMENT_CREATION

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`DEFERRED_SEGMENT_CREATION` specifies the semantics of deferred segment creation. If set to `true`, then segments for tables and their dependent objects (LOBs, indexes) will not be created until the first row is inserted into the table.

Before creating a set of tables, if it is known that a significant number of them will not be populated, then consider setting this parameter to `true`. This saves disk space and minimizes install time.

# DG_BROKER_CONFIG_FILE*n*

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DG_BROKER_CONFIG_FILE[1 \| 2] = filename` |
| **Default value** | Operating system-dependent |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | One filename |
| **Basic** | No |

`DG_BROKER_CONFIG_FILE`*n* (where *n* = 1, 2) specifies the names for the Data Guard broker configuration files.

Every database that is part of a Data Guard broker configuration has two broker configuration files, which contain entries that describe the state and properties of the configuration (such as the sites and databases that are part of the configuration, the roles and properties of each of the databases, and the state of each of the elements of the configuration). Two files are provided to always maintain the last known good state of the configuration.

If `DG_BROKER_CONFIG_FILE`*n* is not explicitly defined, then it is set to an operating system-specific default value at instance startup. The parameter can only be altered when the DMON (Data Guard broker) process is not running.

**See Also:**

*Oracle Data Guard Broker* for more information about setting this parameter

# DG_BROKER_START

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`DG_BROKER_START` enables Oracle to determine whether the Oracle Data Guard broker (DMON) process should be started. DMON is a non-fatal Oracle background process and exists as long as the instance exists, whenever this parameter is set to `true`.

If the site is never going to be configured in an Oracle Data Guard broker configuration, then you can leave the parameter unspecified and accept the default value of `false`. If the site is part of an Oracle Data Guard broker configuration, then automatic startup of the DMON process is simplified by setting `DG_BROKER_START` to `true` in the initialization parameter file.

**See Also:**

*Oracle Data Guard Broker* for examples of using this parameter to start the Oracle Data Guard broker

# DIAGNOSTIC_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DIAGNOSTIC_DEST = { pathname \| directory }` |
| **Default value** | Derived from the value of the `$ORACLE_BASE` environment variable. If `$ORACLE_BASE` is not set, then derived from `ORACLE_BASE` as set by the Oracle Universal Installer. If `ORACLE_BASE` is not set, then `$ORACLE_HOME/rdbms/log` is used. |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | This parameter can be set on each instance. Oracle recommends that each instance in a cluster specify a `DIAGNOSTIC_DEST` directory location that is located on shared disk and that the same value for `DIAGNOSTIC_DEST` be specified for each instance. |

As of Oracle Database 11g Release 1 (11.1), the diagnostics for each database instance are located in a dedicated directory, which can be specified through the `DIAGNOSTIC_DEST` initialization parameter. The structure of the directory specified by `DIAGNOSTIC_DEST` is as follows:

<*diagnostic_dest*>/diag/rdbms/<*dbname*>/<*instname*>

This location is known as the Automatic Diagnostic Repository (ADR) Home. For example, if the database name is `proddb` and the instance name is `proddb1`, the ADR home directory would be `<diagnostic_dest>/diag/rdbms/proddb/proddb1`.

The following files are located under the ADR home directory:

- Trace files - located in subdirectory `<adr-home>/trace`
- Alert logs - located in subdirectory `<adr-home>/alert`. In addition, the `alert.log` file is now in XML format, which conforms to the Oracle ARB logging standard.
- Core files - located in the subdirectory `<adr-home>/cdump`
- Incident files - the occurrence of each serious error (for example, ORA-600, ORA-1578, ORA-7445) causes an incident to be created. Each incident is assigned an ID and dumping for each incident (error stack, call stack, block dumps, and so on) is stored in its own file, separated from process trace files. Incident dump files are located in `<adr-home>/incident/<incdir#>`. You can find the incident dump file location inside the process trace file.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for an example of the diagnostic directory for an Oracle ASM instance

# DISK_ASYNCH_IO

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`DISK_ASYNCH_IO` controls whether I/O to datafiles, control files, and logfiles is asynchronous (that is, whether parallel server processes can overlap I/O requests with CPU processing during table scans). If your platform supports asynchronous I/O to disk, Oracle recommends that you leave this parameter set to its default value. However, if the asynchronous I/O implementation is not stable, you can set this parameter to `false` to disable asynchronous I/O. If your platform does not support asynchronous I/O to disk, this parameter has no effect.

If you set `DISK_ASYNCH_IO` to `false`, then you can increase `DB_WRITER_PROCESSES` or use `DBWR_IO_SLAVES` to simulate asynchronous I/O.

**See Also:**

- "DB_WRITER_PROCESSES"
- "DBWR_IO_SLAVES"
- *Oracle Database Performance Tuning Guide* for more information about choosing between multiple DBWR processes and I/O slaves

# DISPATCHERS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `DISPATCHERS = '`*`dispatch_clause`*`'`<br>**`dispatch_clause::=`**<br>`(PROTOCOL =` *`protocol`*`) \|`<br>`(ADDRESS =` *`address`*`) \|`<br>`(DESCRIPTION =` *`description`* `)`<br>`[options_clause]`<br>**`options_clause::=`**<br>`(DISPATCHERS =` *`integer`* `\|`<br>`SESSIONS =` *`integer`* `\|`<br>`CONNECTIONS =` *`integer`* `\|`<br>`MULTIPLEX = {1 \| ON \| YES \| TRUE \|`<br>`0 \| OFF \| NO \| FALSE \| BOTH \| IN \| OUT} \|`<br>`LISTENER =` *`tnsname`* `\|`<br>`SERVICE =` *`service`* `\|`<br>`INDEX =` *`integer`*`)` |
| **Default value** | If `SHARED_SERVERS` is greater than `0`, then `DISPATCHERS` defaults to `'(PROTOCOL=tcp)'`, causing 1 TCP/IP dispatcher to be created. |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`DISPATCHERS` configures dispatcher processes in the shared server architecture. The parsing software supports a name-value syntax to enable the specification of attributes in a position-independent, case-insensitive manner. For example:

```
DISPATCHERS = '(PROTOCOL=TCP)(DISPATCHERS=3)'
```

Attributes may be specified using the full attribute name or any substring beginning with the first 3 characters. For example, `SESSIONS` can be specified as `SES`, `SESS`, `SESSI`, and so on.

Specify only one of the following attributes: `PROTOCOL`, `ADDRESS`, or `DESCRIPTION`. If you specify either `ADDRESS` or `DESCRIPTION`, then you can specify additional network attributes. Doing so supports multi-homed hosts.

**dispatch_clause**

- PROTOCOL

  The network protocol for which the dispatcher generates a listening endpoint.

- ADDRESS

  The network protocol address of the endpoint on which the dispatchers listen.

- DESCRIPTION

  The network description of the endpoint on which the dispatchers listen, including the protocol address.

**options_clause**

- DISPATCHERS

  The initial number of dispatchers to start. The default is 1.

- SESSIONS

  The maximum number of network sessions to allow for each dispatcher. The default is operating system-specific. Most operating systems have a default of 16 KB.

- CONNECTIONS

  The maximum number of network connections to allow for each dispatcher. The default is operating system-specific.

- MULTIPLEX

  Enables the Oracle Connection Manager session multiplexing feature.

  – The values 1, ON, YES, TRUE, and BOTH indicate that Network Session Multiplex is enabled for both incoming and outgoing network connections.

  – The value IN indicates that Network Session Multiplex is enabled for incoming network connections.

  – The value OUT indicates that Network Session Multiplexing is enabled for outgoing network connections.

  – The values 0, NO, OFF, and FALSE indicate that Network Session Multiplexing is disabled for both incoming and outgoing network connections. This is the default.

- LISTENER

  Specifies the network name of an address or address list of the Oracle Net listeners with which the dispatchers will register.

  The LISTENER attribute facilitates administration of multi-homed hosts. This attribute specifies the appropriate listeners with which the dispatchers will register. The LISTENER attribute takes precedence over the LOCAL_LISTENER and REMOTE_LISTENER parameters. See "LOCAL_LISTENER" and "REMOTE_LISTENER".

- SERVICE

Specifies one or more names by which clients can connect to the dispatchers. The SERVICE attribute takes precedence over the SERVICE_NAMES parameter.

- INDEX

  Use this attribute in an ALTER SYSTEM SET DISPATCHERS statement to indicate which dispatcher configuration you want to modify. (If you specify INDEX in the initialization parameter file, the Oracle Database ignores it.) In an ALTER SYSTEM statement, INDEX specifies the order in which the parameter's values were initialized. The value ranges from 0 (for the first dispatcher configuration) to one less than the total number of dispatcher configurations you define.

  For example, if you specify 3 dispatcher configurations in the initialization parameter file, you would modify the third dispatcher configuration by specifying INDEX=2 in the ALTER SYSTEM statement. You could also add another dispatcher configuration in the ALTER SYSTEM statement by specifying INDEX=3.

  If INDEX is not specified in the ALTER SYSTEM statement, then the PROTOCOL, ADDRESS, or DESCRIPTION attributes must be specified, and if a dispatcher configuration matching this PROTOCOL, ADDRESS, or DESCRIPTION exists, then that configuration will be modified. Otherwise, a new configuration will be added.

**See Also:**

- "SHARED_SERVERS"
- *Oracle Database Net Services Administrator's Guide* and *Oracle Database Administrator's Guide* for more information on setting this parameter

# DISTRIBUTED_LOCK_TIMEOUT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | No |
| **Range of values** | 1 to unlimited |
| **Basic** | No |

DISTRIBUTED_LOCK_TIMEOUT specifies the amount of time (in seconds) for distributed transactions to wait for locked resources.

**See Also:**

*Oracle Database Concepts* for more information on data concurrency

# DML_LOCKS

| Property | Description |
|---|---|
| **Parameter type** | Integer |

| Property | Description |
|---|---|
| **Default value** | Derived: 4 * TRANSACTIONS |
| **Modifiable** | No |
| **Range of values** | 20 to unlimited; a setting of 0 disables enqueues |
| **Basic** | No |
| **Oracle RAC** | You must set this parameter for every instance, and all instances must have positive values or all must be 0. |

A **DML lock** is a lock obtained on a table that is undergoing a DML operation (insert, update, delete). DML_LOCKS specifies the maximum number of DML locks—one for each table modified in a transaction. The value should equal the grand total of locks on tables currently referenced by all users. For example, if three users are modifying data in one table, then three entries would be required. If three users are modifying data in two tables, then six entries would be required.

The default value assumes an average of four tables referenced for each transaction. For some systems, this value may not be enough.

**Enqueues** are shared memory structures that serialize access to database resources. If you set the value of DML_LOCKS to 0, enqueues are disabled and performance is slightly increased. However, you should be aware of the following restrictions when you set you DML_LOCKS to 0:

- You cannot use DROP TABLE, CREATE INDEX statements
- You cannot use explicit lock statements such as LOCK TABLE IN EXCLUSIVE MODE
- Enterprise Manager cannot run on any instances for which DML_LOCKS is set to 0

Oracle holds more locks during parallel DML than during serial execution. Therefore, if your database supports a lot of parallel DML, you may need to increase the value of this parameter.

**See Also:**

- *Oracle Database Concepts* for a discussion of lock and enqueue resources needed for parallel DML
- *Oracle Database Concepts* for more information on data concurrency

# DNFS_BATCH_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | No |
| **Range of values** | 0 - 4096 |

| Property | Description |
|---|---|
| **Basic** | No |

DNFS_BATCH_SIZE controls the number of asynchronous I/O's that can be queued by an Oracle process when Direct NFS Client is enabled. In environments where the NFS server cannot handle a large number of outstanding asynchronous I/O requests, use this parameter to limit the number of I/O's issued by an Oracle foreground process. The recommended setting for this parameter is to start at 128 and increase or decrease it based on NFS server performance.

**See Also:**

*Oracle Database Performance Tuning Guide* for additional information about the DNFS_BATCH_SIZE initialization parameter

# DST_UPGRADE_INSERT_CONV

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

DST_UPGRADE_INSERT_CONV specifies whether internal operators will be allocated on top of TIMESTAMP WITH TIME ZONE (TSTZ) columns of tables which have not been upgraded during the upgrade window of daylight saving time patching for TIMESTAMP WITH TIME ZONE data.

### Values

- `true`

  Internal operators will be allocated on top of TSTZ columns of tables which have not been upgraded. This is the default.

- `false`

  Internal operators will not be allocated on top of TSTZ columns of tables which have not been upgraded.

When DST_UPGRADE_INSERT_CONV is set to true during the upgrade window of the daylight saving time patching process:

- SELECT queries on tables with TSTZ data which have not been upgraded will use internal operators on top of TSTZ columns to present TSTZ data as if they were recorded using the new time zone translation rules.
- DML on tables with TSTZ data which have not been upgraded will use internal operators on top of TSTZ columns to ensure that the TSTZ data is recorded using

the old time zone translation rules in order to be consistent with the existing TSTZ data in the same tables.

**Note:**

Oracle strongly recommends that this parameter is set to `true` throughout the upgrade window of the daylight saving time patching process. This parameter reduces the performance impact since indexes on TSTZ columns will be disabled whenever internal operators are allocated. If the parameter is set to `false`, then indexes will be used and this may affect performance of queries against TSTZ data during the DST upgrade window. Turning off this parameter during the upgrade window may corrupt data on disk when DMLs occur for tables with TSTZ data which have not yet been upgraded.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

# ENABLE_DDL_LOGGING

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`ENABLE_DDL_LOGGING` enables or disables the writing of a subset of data definition language (DDL) statements to a DDL log.

The DDL log is a file that has the same format and basic behavior as the alert log, but it only contains the DDL statements issued by the database. The DDL log is created only for the RDBMS component and only if the `ENABLE_DDL_LOGGING` initialization parameter is set to `true`. When this parameter is set to `false`, DDL statements are not included in any log.

The DDL log contains one log record for each DDL statement issued by the database. The DDL log is included in IPS incident packages.

There are two DDL logs that contain the same information. One is an XML file, and the other is a text file. The DDL log is stored in the `log/ddl` subdirectory of the ADR home.

When `ENABLE_DDL_LOGGING` is set to `true`, the following DDL statements are written to the log:

- `ALTER/CREATE/DROP/TRUNCATE CLUSTER`
- `ALTER/CREATE/DROP FUNCTION`

- ALTER/CREATE/DROP INDEX
- ALTER/CREATE/DROP OUTLINE
- ALTER/CREATE/DROP PACKAGE
- ALTER/CREATE/DROP PACKAGE BODY
- ALTER/CREATE/DROP PROCEDURE
- ALTER/CREATE/DROP PROFILE
- ALTER/CREATE/DROP SEQUENCE
- CREATE/DROP SYNONYM
- ALTER/CREATE/DROP/RENAME/TRUNCATE TABLE
- ALTER/CREATE/DROP TRIGGER
- ALTER/CREATE/DROP TYPE
- ALTER/CREATE/DROP TYPE BODY
- DROP USER
- ALTER/CREATE/DROP VIEW

**Note:**

The DDL statement written to the log may be truncated. You can use DDL triggers to view the entire DDL statement. See *Oracle Database PL/SQL Language Reference* for more information about DDL triggers.

**See Also:**

*Oracle Database Licensing Information* for licensing information for the ENABLE_DDL_LOGGING initialization parameter.

# ENABLE_GOLDENGATE_REPLICATION

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | true |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | true \| false |
| **Basic** | No |
| **Oracle RAC** | All instances must have the same setting |

ENABLE_GOLDENGATE_REPLICATION controls services provided by the RDBMS for Oracle GoldenGate (both capture and apply services). Set this to true to enable RDBMS services used by Oracle GoldenGate.

This parameter primarily controls supplemental logging required to support logical replication of new data types and operations. The redo log file is designed to be applied physically to a database, therefore the default contents of the redo log file often do not contain sufficient information to allow logged changes to be converted into SQL statements. Supplemental logging adds extra information into the redo log files so that replication can convert logged changes into SQL statements without having to access the database for each change. Previously these extra changes were controlled by the supplemental logging DDL. Now the ENABLE_GOLDENGATE_REPLICATION parameter must also be set to enable the required supplemental logging for any new data types or operations.

All enhancements to supplemental logging required to support logical replication are also controlled by this parameter.

The RDBMS services controlled by this parameter also include (but are not limited to):

- Transparent Data Encryption (including Tablespace Encryption) utilities used by GoldenGate Extract
- Service to read redo logs used by GoldenGate Extract
- Service to suppress triggers used by GoldenGate Replicat
- Service to handle transient duplicate handling used by GoldenGate Replicat
- Service to bypass referential integrity checking used by GoldenGate Replicat
- Services required to run Oracle GoldenGate in Integrated Extract and Integrated Replicat

**See Also:**

*Oracle GoldenGate Installing and Configuring Oracle GoldenGate for Oracle Database* for more information about the ENABLE_GOLDENGATE_REPLICATION initialization parameter

# ENABLE_PLUGGABLE_DATABASE

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | No |
| **Range of values** | true \| false |
| **Basic** | No |

ENABLE_PLUGGABLE_DATABASE is a bootstrap initialization parameter to create a CDB, and it enables a database in NOMOUNT startup mode to know that it is a CDB.

This parameter must be set in init.ora before creating a CDB.

**Note:**

See *Oracle Database Concepts* for more information about CDBs and PDBs.

# EVENT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Basic** | No |

EVENT is a parameter used only to debug the system. Do not alter the value of this parameter except under the supervision of Oracle Support Services staff.

# EXAFUSION_ENABLED

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | No |
| **Range of values** | 0, 1 |
| **Basic** | No |
| **Oracle RAC** | All of the instances in an Oracle RAC cluster must enable this parameter, or all of the instances in an Oracle RAC cluster must disable the parameter. |

EXAFUSION_ENABLED is used to enable or disable the Exafusion accelerate cache fusion protocol feature, which is available on the Linux operating system of Oracle Exadata Database Machine only. Exafusion provides lower latency, higher throughput direct to wire block transfers between Oracle Real Application Clusters (Oracle RAC) instances. Data is transferred directly from user space to the Infiniband network, leading to reduced CPU utilization and better scale-out performance.

Exafusion is disabled by default. To enable Exafusion, set the EXAFUSION_ENABLED initialization parameter to 1. To disable Exafusion, set the EXAFUSION_ENABLED initialization parameter to 0. This parameter cannot be set dynamically. It must be set before instance startup.

These are the prerequisites before enabling Exafusion:

- Oracle Storage Server Software release 12.1.2.1.1 or later is required.
- Mellanox ConnectX-2 and ConnectX-3 Host Channel Adapters (HCAs) are required.

- Oracle Unbreakable Enterprise Kernel 2 Quarterly Update 5 (UEK2QU5) kernels (2.6.39-400.2nn) or later are required.

The Exafusion feature does not support mixed mode operation on an Oracle RAC cluster. Either all the instances in an Oracle RAC cluster must have the feature enabled or none. Rolling upgrades of a non-Exafusion enabled Oracle RAC cluster to Exafusion are not supported.

**Note:**

Exafusion is available only on the Linux operating system of Oracle Exadata Database Machine. It is not supported on any non-engineered system or SPARC SuperCluster (SSC). If you set the `EXAFUSION_ENABLED` parameter on a non-supported platform, then the instance will be prevented from starting up, and an ORA-00439: `The specified feature is not enabled:` *string* error message is generated.

**Note:**

This initialization parameter is available starting with Oracle Database 12c Release 1 (12.1.0.2) Bundle Patch 6 (BP6).

## FAL_CLIENT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `FAL_CLIENT = string` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`FAL_CLIENT` specifies the FAL (fetch archive log) client name that is used by the FAL service, configured through the `FAL_SERVER` initialization parameter, to refer to the FAL client. The value is an Oracle Net service name, which is assumed to be configured properly on the FAL server system to point to the FAL client (standby database).

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about FAL server

## FAL_SERVER

| Property | Description |
|---|---|
| **Parameter type** | String |

| Property | Description |
|---|---|
| **Syntax** | `FAL_SERVER = string` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

FAL_SERVER specifies the FAL (fetch archive log) server for a standby database. The value is an Oracle Net service name, which is assumed to be configured properly on the standby database system to point to the desired FAL server.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about FAL server

# FAST_START_MTTR_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 0 to 3600 seconds |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values, and you can change the values at run time. |

FAST_START_MTTR_TARGET enables you to specify the number of seconds the database takes to perform crash recovery of a single instance. When specified, FAST_START_MTTR_TARGET is overridden by LOG_CHECKPOINT_INTERVAL.

**See Also:**

- *Oracle Database Performance Tuning Guide* for an example of setting the lower bound for this parameter
- *Oracle Database Performance Tuning Guide* for an example of setting the upper bound for this parameter
- *Oracle Database Performance Tuning Guide* for an example of selecting a preliminary value for this parameter

# FAST_START_PARALLEL_ROLLBACK

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `FAST_START_PARALLEL_ROLLBACK = { HIGH \| LOW \| FALSE }` |
| **Default value** | `LOW` |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`FAST_START_PARALLEL_ROLLBACK` specifies the degree of parallelism used when recovering terminated transactions. Terminated transactions are transactions that are active before a system failure. If a system fails when there are uncommitted parallel DML or DDL transactions, then you can speed up transaction recovery during startup by using this parameter.

### Values

- `FALSE`

  Parallel rollback is disabled

- `LOW`

  Limits the maximum degree of parallelism to 2 * `CPU_COUNT`

- `HIGH`

  Limits the maximum degree of parallelism to 4 * `CPU_COUNT`

If you change the value of this parameter, then transaction recovery will be stopped and restarted with the new implied degree of parallelism.

**See Also:**

*Oracle Database VLDB and Partitioning Guide* for more information about this parameter

## FILE_MAPPING

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

**Note:**

The FILE_MAPPING initialization parameter is deprecated. It is still supported for backward compatibility.

FILE_MAPPING specifies whether file mapping is enabled. The FMON background process is started to manage the mapping information when file mapping is enabled.

**See Also:**

*Oracle Database Administrator's Guide* for more information about the file mapping feature

# FILEIO_NETWORK_ADAPTERS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | FILEIO_NETWORK_ADAPTERS = *adapter_name* [, *adapter_name* ] ... |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | One or more network adapter names, separated by commas |
| **Basic** | No |

FILEIO_NETWORK_ADAPTERS specifies a list of network adapters that can be used to access the disk storage. On platforms where the database files reside in network attached storage, this parameter provides the storage access library the list of network adapters that can be used to access the storage.

The network adapter name is a fully qualified address name of the network card that can be accessed through the host name database or using the Network Information Service. The components of the adapter name are separated by periods. For example, the following is a fully qualified adapter name:

```
ib1.oracle.com
```

# FILESYSTEMIO_OPTIONS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | FILESYSTEMIO_OPTIONS = { none \| setall \| directIO \| asynch } |
| **Default value** | Varies by database version and operating system. |
| **Modifiable** | No |
| **Basic** | No |

FILESYSTEMIO_OPTIONS specifies I/O operations for file system files.

**See Also:**

*Oracle Database Performance Tuning Guide* for information about setting this parameter before running I/O calibration

# FIXED_DATE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | FIXED_DATE = [YYYY-MM-DD-HH24:MI:SS (or the default Oracle date format) \| NONE] |
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

FIXED_DATE enables you to set a constant date that SYSDATE will always return instead of the current date. To undo a fixed date setting, specify FIXED_DATE=NONE. This parameter is useful primarily for testing. The value can be in the format shown above or in the default Oracle date format, without a time.

# GCS_SERVER_PROCESSES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | If 1 - 3 CPUS, then 1<br>If 4 - 15 CPUs, then 2<br>If 16 or more CPUs, then 2 + (CPUs / 32). If the result includes a fraction, then the fraction is disregarded. For example, if you had 20 CPUs, then 2 + (20 / 32) would equal 2 GCS processes.[1]<br>If CLUSTER_DATABASE is set to false, then 0<br>If Oracle ASM, then 1 |
| **Modifiable** | No |
| **Range of values** | 1 to 36 |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

[1] On certain operating systems, the RDBMS optimizes the default allocation of background GCS server processes based on the core to thread ratio of the CPU.

GCS_SERVER_PROCESSES specifies the number of background GCS server processes (LMS0, ... LMS9 and LMSa, ... LMSz) to serve the inter-instance traffic among Oracle RAC instances. GCS server processes are only seen in an Oracle RAC environment.

# GLOBAL_CONTEXT_POOL_SIZE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Default value** | `NULL` |
| **Modifiable** | No |
| **Range of values** | Any integer value in MB |
| **Basic** | No |

**Note:**

The `GLOBAL_CONTEXT_POOL_SIZE` parameter is deprecated. It is retained for backward compatibility only.

`GLOBAL_CONTEXT_POOL_SIZE` specifies the amount of memory to allocate in the SGA for storing and managing global application context.

# GLOBAL_NAMES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true | false` |
| **Basic** | No |

`GLOBAL_NAMES` specifies whether a database link is required to have the same name as the database to which it connects.

If the value of `GLOBAL_NAMES` is `false`, then no check is performed. If you use or plan to use distributed processing, then Oracle recommends that you set this parameter to `true` to ensure the use of consistent naming conventions for databases and links in a networked environment.

**See Also:**

*Oracle Database Administrator's Guide* for more information on setting this parameter

# GLOBAL_TXN_PROCESSES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `1` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `0` to `20` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`GLOBAL_TXN_PROCESSES` specifies the initial number of GTX*n* background processes (GTX0, ... GTX9 and GTXa, ... GTXj) per instance to support global (XA) transactions in an Oracle RAC environment.

If you want to disable the GTX*n* background processes, then you must set `GLOBAL_TXN_PROCESSES` to `0` in your parameter file. Setting this parameter to `0` will disable the XA support on an Oracle RAC database. Error `ORA-55712` will be returned if you try to run XA transactions on an Oracle RAC database with this parameter set to zer`0`. You can change the setting to a nonzero value at run time to turn on the support for XA.

`GLOBAL_TXN_PROCESSES` is useful for systems that process global (XA) transactions heavily. You do not need to specify a value for this parameter since Oracle Database automatically determines the number of processes and autotunes them, as necessary. GTX*n* background processes are only seen in an Oracle RAC environment.

**See Also:**

*Oracle Database Development Guide* for more information about this parameter

# HASH_AREA_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | Derived: 2 * `SORT_AREA_SIZE` |
| **Modifiable** | `ALTER SESSION` |
| **Range of values** | 0 to operating system-dependent |
| **Basic** | No |

**Note:**

Oracle does not recommend using the HASH_AREA_SIZE parameter unless the instance is configured with the shared server option. Oracle recommends that you enable automatic sizing of SQL working areas by setting PGA_AGGREGATE_TARGET instead. HASH_AREA_SIZE is retained for backward compatibility.

HASH_AREA_SIZE is relevant to parallel execution operations and to the query portion of DML or DDL statements. It specifies the maximum amount of memory, in bytes, to be used for hash joins.

**See Also:**

- *Oracle Database Concepts* for information on hash joins in general

# HEAT_MAP

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | HEAT_MAP = { ON \| OFF } |
| **Default value** | OFF |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |
| **Oracle RAC** | If specified, use the same value on all instances |

Setting the HEAT_MAP initialization parameter to ON causes the database to track read and write access of all segments, as well as modification of database blocks, due to DMLs and DDLs. The activities are not tracked for objects in SYSTEM and SYSAUX tablespaces.

Use this initialization parameter to enable or disable both the Heat Map and Automatic Data Optimization (ADO) features.

**See Also:**

- *Oracle Database VLDB and Partitioning Guide* for more information about enabling and disabling the Heat Map feature
- *Oracle Database VLDB and Partitioning Guide* for more information about Heat Map tracking

## HI_SHARED_MEMORY_ADDRESS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | No |
| **Basic** | No |

`HI_SHARED_MEMORY_ADDRESS` specifies the starting address at run time of the system global area (SGA). It is ignored on platforms that specify the SGA's starting address at linktime.

On 64-bit platforms, use `HI_SHARED_MEMORY_ADDRESS` to specify the high-order 32 bits of a 64-bit address. Use `SHARED_MEMORY_ADDRESS` to specify the low-order 32 bits of the address (see "SHARED_MEMORY_ADDRESS"). If both parameters are 0 or unspecified, the SGA address defaults to a platform-specific location.

## HS_AUTOREGISTER

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`HS_AUTOREGISTER` enables or disables automatic self-registration of Heterogeneous Services (HS) agents. When enabled, information is uploaded into the server's data dictionary to describe a previously unknown agent class or a new agent version.

Oracle recommends that you set this parameter to `true`. Oracle incurs less overhead when establishing subsequent connections through the same agent if self-registered information is available in the server's data dictionary.

**See Also:**

*Oracle Database Heterogeneous Connectivity User's Guide* for more information about using HS agents

## IFILE

| Property | Description |
|---|---|
| **Parameter type** | Parameter file |
| **Syntax** | `IFILE = parameter_file_name` |

| Property | Description |
|---|---|
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | Valid parameter filenames |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

Use `IFILE` to embed another parameter file within the current parameter file. For example:

```
IFILE = COMMON.ORA
```

You can have up to three levels of nesting. In this example, the file `COMMON.ORA` could contain a second `IFILE` parameter for the file `COMMON2.ORA`, which could contain a third `IFILE` parameter for the file `GCPARMS.ORA`. You can also include multiple parameter files in one parameter file by listing `IFILE` several times with different values:

```
IFILE = DBPARMS.ORA
IFILE = GCPARMS.ORA
IFILE = LOGPARMS.ORA
```

**Note:**

You must list multiple entries on contiguous lines of the parameter file.

# INMEMORY_CLAUSE_DEFAULT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `INMEMORY_CLAUSE_DEFAULT = '[INMEMORY] [NO INMEMORY] [other-clauses]'`<br>**other-clauses::=**<br>`[compression-clause] [priority-clause] [rac-clause]`<br>**compression-clause::=**<br>`NO MEMCOMPRESS \| MEMCOMPRESS FOR { DML \| QUERY [ LOW \| HIGH ] \| CAPACITY [LOW \| HIGH] } *`<br>**priority-clause::=**<br>`PRIORITY { LOW \| MEDIUM \| HIGH \| CRITICAL \| NONE } *`<br>**rac-clause::=**<br>`[distribute-clause] [duplicate-clause] *`<br>**distribute-clause::=**<br>`DISTRIBUTE [ AUTO \| BY ROWID RANGE \| BY PARTITION \| BY SUBPARTITION ] *` |

| Property | Description |
|---|---|
| | **duplicate-clause::=**<br>`NO DUPLICATE \| DUPLICATE [ ALL ] *` |
| **Default value** | An empty string |
| **Modifiable** | `ALTER SESSION, ALTER SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | All instances should use the same value |

*: See the "Meaning of INMEMORY_CLAUSE_DEFAULT Parameter Values" table in this topic for more information about this clause.

`INMEMORY_CLAUSE_DEFAULT` enables you to specify a default In-Memory Column Store (IM column store) clause for new tables and materialized views.

If the `INMEMORY_CLAUSE_DEFAULT` parameter is unset or set to an empty string (the default), only tables and materialized views explicitly specified as `INMEMORY` will be populated into the IM column store. Setting the value of the `INMEMORY_CLAUSE_DEFAULT` parameter to `NO INMEMORY` has the same effect as setting it to the default value.

If the `INMEMORY_CLAUSE_DEFAULT` parameter is set, then any *newly created* table or materialized view specified as `INMEMORY` will inherit unspecified attributes from this parameter. This can force certain in-memory options by default that are not explicitly specified in the syntax. For example, if the `INMEMORY_CLAUSE_DEFAULT` parameter is set to `MEMCOMPRESS FOR CAPACITY LOW` and a table is created as `INMEMORY PRIORITY HIGH`, then the table is treated as if it was declared as `INMEMORY MEMCOMPRESS FOR CAPACITY LOW PRIORITY HIGH`.

If `INMEMORY` is specified as part of this parameter, then all newly created tables and materialized views will be populated into the IM column store, except tables and materialized views explicitly specified as `NO INMEMORY`. For example, if this parameter is set to `INMEMORY MEMCOMPRESS FOR CAPACITY HIGH`, then all new tables will be created as if this clause were present in the SQL `CREATE TABLE` statement. If there is a default `INMEMORY` value for the tablespace for a given segment, then it will override the value for this parameter.

***Table 1-2 Meaning of INMEMORY_CLAUSE_DEFAULT Parameter Values***

| Syntax | Description |
|---|---|
| `INMEMORY` | Specifies that all newly-created tables and materialized views populate the IM column store unless they are specified as `NO INMEMORY` in the SQL `CREATE TABLE` or `CREATE MATERIALIZED VIEW` statement |
| `NO INMEMORY` | Specifies that only tables and materialized views explicitly specified as `INMEMORY` in the SQL `CREATE TABLE` or `CREATE MATERIALIZED VIEW` statements populate the IM column store |
| `compression-clause` | Specifies that in-memory compression should be used for the instance. Use the `MEMCOMPRESS FOR` values to specify the in-memory compression level. |

| Syntax | Description |
|---|---|
| NO MEMCOMPRESS | When NO MEMCOMPRESS is specified, no in-memory compression is done in the IM column store. |
| MEMCOMPRESS FOR | MEMCOMPRESS FOR is used to indicate the in-memory compression level for the IM column store. |
| DML | When DML is specified, the IM column store is optimized for DML operations, and some lightweight in-memory compression may be done. |
| QUERY | When QUERY is specified, the in-memory compression level is for high performance. If QUERY is specified without LOW or HIGH, it defaults to QUERY LOW. |
| QUERY LOW | When QUERY LOW is specified, the in-memory compression level provides the highest performance. |
| QUERY HIGH | When QUERY HIGH is specified, the in-memory compression level provides a balance between compression and performance, weighted toward performance. |
| CAPACITY | When CAPACITY is specified without LOW or HIGH, it defaults to CAPACITY LOW. |
| CAPACITY LOW | When CAPACITY LOW is specified, the in-memory compression level is a balance between compression and performance, weighted toward capacity. |
| CAPACITY HIGH | When CAPACITY HIGH is specified, the in-memory compression level is for highest capacity. |
| priority-clause | Specifies the priority to use when populating tables in the IM column store. Use the PRIORITY values to specify the priority.<br>By default, the population of a table in the IM column store can be delayed until the database determines it is useful.<br>On database instance startup, tables are populated in priority order. |
| PRIORITY NONE | When PRIORITY NONE is specified, the population of a table in the IM column store can be delayed until the database determines it is useful.<br>This is the default value when no priority is specified. |
| PRIORITY LOW | When PRIORITY LOW is specified for a table or tables, the population of those tables in the IM column store is done before tables that have no priority specified. |
| PRIORITY MEDIUM | When PRIORITY MEDIUM is specified for a table or tables, the population of those tables in the IM column store is done before tables that have no priority and PRIORITY LOW specified. |
| PRIORITY HIGH | When PRIORITY HIGH is specified for a table or tables, the population of those tables in the IM column store is done before tables that have no priority, PRIORITY LOW, and PRIORITY MEDIUM specified. |
| PRIORITY CRITICAL | When PRIORITY CRITICAL is specified for a table or tables, the population of those tables in the IM column store is done |

| Syntax | Description |
|---|---|
| | before tables that have no priority, `PRIORITY LOW`, `PRIORITY MEDIUM`, and `PRIORITY HIGH` specified. |
| `rac-clause` | Specifies how tables in the IM column store will be managed among Oracle Real Application Clusters (Oracle RAC) instances. Use the `distribute-clause` and `duplicate-clause` to specify how tables in the IM store will be managed in Oracle RAC instances.<br>For a non-Oracle RAC database, these settings have no effect, because the whole table or partition has to be on the single instance. |
| `distribute-clause` | Specifies how a table is distributed among Oracle RAC instances. |
| `DISTRIBUTE AUTO` | Specifies that the database will automatically decide how to distribute tables in the IM column store across the Oracle RAC instances based on the type of partitioning and the value of the `duplicate-clause`.<br>`DISTRIBUTE AUTO` is the default, and it is also used when `DISTRIBUTE` is specified by itself. |
| `DISTRIBUTE BY ROWID RANGE` | Specifies that the tables in the IM column store will be distributed by rowid range to different Oracle RAC instances. |
| `DISTRIBUTE BY PARTITION` | Specifies that partitions in the IM column store will be distributed to different Oracle RAC instances. |
| `DISTRIBUTE BY SUBPARTITION` | Specifies that subpartitions in the IM column store will be distributed to different Oracle RAC instances. |
| `duplicate-clause` | Specifies how many copies of each In-Memory Compression Unit (IMCU) of the tables in the IM column store will be spread across all the Oracle RAC instances.<br>**Note:** The `duplicate-clause` is only applicable if you are using Oracle Real Application Clusters (Oracle RAC) on an engineered system. Otherwise, the `duplicate-clause` is ignored and there is only one copy of each IMCU in memory. |
| `NO DUPLICATE` | Data is not duplicated across Oracle RAC instances. This is the default. |
| `DUPLICATE` | Data is duplicated on another Oracle RAC instance, resulting in data existing on a total of two Oracle RAC instances.. |
| `DUPLICATE ALL` | Data is duplicated across all Oracle RAC instances. If you specify `DUPLICATE ALL`, then the database uses the `DISTRIBUTE AUTO` setting, regardless of whether or how you specify the `distribute-clause`. |

### Examples

The following statement causes no tables to populate the IM column store:

```
alter system set INMEMORY_CLAUSE_DEFAULT='NO INMEMORY'
scope=spfile;
```

The following statement causes new tables and materialized views (except those specified as `NO INMEMORY`) to populate the IM column store at the high capacity compression level:

```
alter system set
INMEMORY_CLAUSE_DEFAULT='INMEMORY MEMCOMPRESS FOR CAPACITY HIGH'
scope=spfile;
```

The following statement causes new tables (even those specified as `NO INMEMORY`) to populate the IM column store at the highest performance compression level at `LOW` priority:

```
alter system set
INMEMORY_CLAUSE_DEFAULT='INMEMORY MEMCOMPRESS FOR QUERY LOW PRIORITY LOW'
scope=spfile;
```

The following statement causes new tables (even those specified as `NO INMEMORY`) to populate the IM column store without any in-memory compression:

```
alter system set
INMEMORY_CLAUSE_DEFAULT='INMEMORY NO MEMCOMPRESS'
scope=spfile;
```

The following statement causes tables in the IM column store to be duplicated on every Oracle RAC instance, unless on a non-engineered system. For a non-engineered system, the `duplicate-clause` (`DUPLICATE ALL`) will be ignored and tables in the column store will be automatically distributed across the Oracle RAC instance, with only one copy of each IMCU in the tables in the IM column store:

```
alter system set
INMEMORY_CLAUSE_DEFAULT='INMEMORY MEMCOMPRESS FOR QUERY
DISTRIBUTE AUTO DUPLICATE ALL'
scope=spfile;
```

The following statement sets the value of the `INMEMORY_CLAUSE_DEFAULT` parameter back to its default value, the empty string:

```
alter system set
INMEMORY_CLAUSE_DEFAULT=''
scope=spfile;
```

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_FORCE"
- "INMEMORY_QUERY"
- "INMEMORY_SIZE"
- "V$IM_SEGMENTS"
- "V$IM_USER_SEGMENTS"
- *Oracle Database Concepts* for overview information on the IM column store
- *Oracle Database Administrator's Guide* for information on using the IM column store
- *Oracle Database Administrator's Guide* for more information on IM column store compression methods
- *Oracle Database Administrator's Guide* for more information on priority levels for populating a database object in the IM column store
- *Oracle Database SQL Language Reference* for more information on the CREATE TABLE statement
- *Oracle Database SQL Language Reference* for more information on the CREATE MATERIALIZED VIEW statement

# INMEMORY_FORCE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `INMEMORY_FORCE = { DEFAULT \| OFF }` |
| **Default value** | `DEFAULT` |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | All instances should use the same value. |

`INMEMORY_FORCE` allows you to specify whether tables and materialized view that are specified as `INMEMORY` are populated into the In-Memory Column Store (IM column store) or not.

The default value is `DEFAULT`. When this value is in effect, the IM column store is populated only with tables and materialized views specified as `INMEMORY`.

If `OFF` is specified, then even if the IM column store is configured on this instance, no tables or materialized are populated in memory.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "INMEMORY_MAX_POPULATE_SERVERS"
- "INMEMORY_QUERY"
- "INMEMORY_SIZE"
- *Oracle Database Concepts* for overview information on the IM column store
- *Oracle Database Administrator's Guide* for information on using the IM column store

# INMEMORY_MAX_POPULATE_SERVERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | Half the effective CPU thread count or the `PGA_AGGREGATE_TARGET` value divided by 512M, whichever is less. |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 0 to a value based on the number of cores in the system. |
| **Basic** | No |
| **Oracle RAC** | All instances should use the same value. |

`INMEMORY_MAX_POPULATE_SERVERS` specifies the maximum number of background populate servers to use for In-Memory Column Store (IM column store) population, so that these servers do not overload the rest of the system.

This parameter has meaning only if the `INMEMORY_SIZE` parameter is also set to a positive value.

The value to use for this parameter depends on the number of cores in the system. A certain percentage of CPU should be allocated for in-memory background population, and this parameter should be set accordingly. You can also set this parameter to `0` to temporarily disable populate tasks on the system from executing.

**Note:**

Be careful not to set the value of this parameter too high. If it is set close to the number of cores or higher, no CPU could be left for the rest of the system to run.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "INMEMORY_FORCE"
- "INMEMORY_QUERY"
- "INMEMORY_SIZE"
- *Oracle Database Concepts* for overview information on the IM column store
- *Oracle Database Administrator's Guide* for information on using the IM column store

# INMEMORY_QUERY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `INMEMORY_QUERY = { ENABLE \| DISABLE }` |
| **Default value** | `ENABLE` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | All instances should use the same value. |

`INMEMORY_QUERY` is used to enable or disable in-memory queries for the entire database at the session or system level. This parameter is helpful when you want to test workloads with and without the use of the In-Memory Column Store (IM column store).

This parameter enables in-memory queries for the entire database by default when the `INMEMORY_SIZE` parameter is specified.

Set this parameter to `DISABLE` if you want to disable in-memory queries.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "INMEMORY_FORCE"
- "INMEMORY_MAX_POPULATE_SERVERS"
- "INMEMORY_SIZE"
- *Oracle Database Concepts* for overview information on the IM column store
- *Oracle Database Administrator's Guide* for information on using the IM column store

# INMEMORY_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `INMEMORY_SIZE = integer [K \| M \| G]` |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 0 to the amount of memory left in the SGA after other allocations |
| **Basic** | No |
| **Oracle RAC** | All instances should use the same value. |

`INMEMORY_SIZE` sets the size of the In-Memory Column Store (IM column store) on a database instance.

The default value is 0, which means that the IM column store is not used.

The database must be restarted after setting this parameter to enable the IM column store.

The minimum size to which this parameter can be set is 100 MB.

Typically this parameter should be set to at least the size needed to accommodate all the tables that will use the IM column store. It can be set higher to allow for growth of those tables or to accommodate other tables that will use the IM column store in the future.

This parameter can also be set per PDB to limit the maximum size of the IM column store for that PDB. Note that the sum of the PDB values do not have to equal the CDB value, and the sum of the PDB values may even be greater than the CDB value.

Unless this parameter is specifically set on a PDB, each PDB inherits the CDB value, which means they can use all of the available IM column store.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "INMEMORY_FORCE"
- "INMEMORY_MAX_POPULATE_SERVERS"
- "INMEMORY_QUERY"
- *Oracle Database Concepts* for overview information on the IM column store
- *Oracle Database Administrator's Guide* for information on using the IM column store
- *Oracle Database Administrator's Guide* for an example of using the `INMEMORY_SIZE` parameter

# INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `1` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `0 to 50` |
| **Basic** | No |
| **Oracle RAC** | All instances should use the same value. |

`INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT` limits the maximum number of background populate servers used for In-Memory Column Store (IM column store) repopulation, as trickle repopulation is designed to use only a small percentage of the populate servers. The value for this parameter is a percentage of the `INMEMORY_MAX_POPULATE_SERVERS` initialization parameter value.

For example, if this parameter is set to 5 and INMEMORY_MAX_POPULATE_SERVERS is set to 10, then on average half of a core is used for trickle repopulation.

The default value of 1 is good in most cases. In some cases, if you want to disable trickle repopulate, this parameter can be set to 0. If you want to keep the system more aggressively up to date (at the expense of more background CPU), you can set the parameter to higher values such as 5 or 10.

A value of greater than 50 is not allowed, so that at least half of the populate servers are available for other (re)populate tasks. On some systems, a value of less than 50 can be problematic, depending on other workload.

This parameter has meaning only if the INMEMORY_SIZE parameter is also set to a positive value.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# INSTANCE_GROUPS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | INSTANCE_GROUPS = *group_name* [, *group_name* ] ... |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | One or more instance group names, separated by commas |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

**Note:**

The INSTANCE_GROUPS parameter is deprecated. It is retained for backward compatibility only.

INSTANCE_GROUPS is an Oracle RAC parameter that you can specify only in parallel mode. Used with the PARALLEL_INSTANCE_GROUP parameter, it lets you restrict parallel query operations to a limited number of instances.

This parameter specifies one or more instance groups and assigns the current instance to those groups. If one of the specified groups is also specified in the PARALLEL_INSTANCE_GROUP parameter, then Oracle allocates query processes for a parallel operation from this instance.

**See Also:**

- *Oracle Real Application Clusters Administration and Deployment Guide* for more information on parallel query execution in a Real Application Clusters environment
- "PARALLEL_INSTANCE_GROUP"

# INSTANCE_NAME

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `INSTANCE_NAME = instance_id` |
| **Default value** | The instance's SID<br>**Note:** The SID identifies the instance's shared memory on a host, but may not uniquely distinguish this instance from other instances. |
| **Modifiable** | No |
| **Range of values** | Alphanumeric ASCII characters and the underscore (_) character, up to a maximum of 255 characters |
| **Basic** | No |

In a Real Application Clusters environment, multiple instances can be associated with a single database service. Clients can override Oracle's connection load balancing by specifying a particular instance by which to connect to the database. `INSTANCE_NAME` specifies the unique name of this instance.

In a single-instance database system, the instance name is usually the same as the database name.

**See Also:**

*Oracle Real Application Clusters Administration and Deployment Guide* and *Oracle Database Net Services Administrator's Guide* for more information

# INSTANCE_NUMBER

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` (zero) |
| **Modifiable** | No |
| **Range of values** | 1 to maximum number of instances specified when the database was created |
| **Basic** | Yes |
| **Oracle RAC** | You must set this parameter for every instance, and all instances must have different values. |

`INSTANCE_NUMBER` is an Oracle RAC parameter that can be specified in parallel mode or exclusive mode. It specifies a unique number that maps the instance to one free list group for each database object created with storage parameter `FREELIST GROUPS`.

The `INSTANCE` parameter of the `ALTER TABLE ... ALLOCATE EXTENT` statement assigns an extent to a particular free list group. If you set `INSTANCE_NUMBER` to the value specified for the `INSTANCE` parameter, the instance uses that extent for inserts and for updates that expand rows.

The practical maximum value of this parameter is the maximum number of instances specified in the `CREATE DATABASE` statement. The absolute maximum is operating system-dependent.

**See Also:**

*Oracle Real Application Clusters Administration and Deployment Guide* for more information

# INSTANCE_TYPE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `INSTANCE_TYPE = { RDBMS \| ASM \| ASMPROXY }` |
| **Default value** | `RDBMS` |
| **Modifiable** | No |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must have the same value. |

`INSTANCE_TYPE` specifies whether the instance is a database instance, an Oracle Automatic Storage Management (Oracle ASM) instance, or an Oracle ASM Proxy instance.

### Values

- `RDBMS`

  The instance is a database instance.

- `ASM`

  The instance is an Oracle ASM instance.

- `ASMPROXY`

  The instance is an Oracle ASM proxy instance.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for information about managing Oracle Flex ASM

# JAVA_JIT_ENABLED

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | Operating system-dependent |

| Property | Description |
|---|---|
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`JAVA_JIT_ENABLED` enables or disables the Just-in-Time (JIT) compiler for the Oracle Java Virtual Machine (OracleJVM) environment.

For platforms that support the JIT compiler, the default value of this parameter is `true`; otherwise the default value is `false`. Attempting to set this parameter to `true` on unsupported platforms will result in an error.

**See Also:**

*Oracle Database Java Developer's Guide* for more information about setting this parameter

## JAVA_MAX_SESSIONSPACE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | No |
| **Range of values** | 0 to 2 GB - 1 |
| **Basic** | No |

Java session space is the memory that holds Java state from one database call to another. `JAVA_MAX_SESSIONSPACE_SIZE` specifies (in bytes) the maximum amount of session space made available to a Java program executing in the server. When a user's session-duration Java state attempts to exceed this amount, the Java virtual machine kills the session with an out-of-memory failure.

**See Also:**

- *Oracle Database Java Developer's Guide* for more information about this parameter
- "JAVA_SOFT_SESSIONSPACE_LIMIT"

## JAVA_POOL_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `JAVA_POOL_SIZE = integer [K \| M \| G]` |

| Property | Description |
|---|---|
| **Default value** | If SGA_TARGET is set: If the parameter is not specified, then the default is 0 (internally determined by the Oracle Database). If the parameter is specified, then the user-specified value indicates a minimum value for the memory pool.<br>If SGA_TARGET is not set: 24 MB, rounded up to the nearest granule size |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | Minimum: 0 (values greater than zero are rounded up to the nearest granule size)<br>Maximum: operating system-dependent |
| **Basic** | No |

JAVA_POOL_SIZE specifies (in bytes) the size of the Java pool, from which the Java memory manager allocates most Java state during run-time execution. This memory includes the shared in-memory representation of Java method and class definitions, as well as the Java objects that are migrated to the Java session space at end-of-call.

**See Also:**

*Oracle Database Java Developer's Guide* for more information about this parameter

# JAVA_SOFT_SESSIONSPACE_LIMIT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | No |
| **Range of values** | 0 to 2 GB - 1 |
| **Basic** | No |

Java session space is the memory that holds Java state from one database call to another. JAVA_SOFT_SESSIONSPACE_LIMIT specifies (in bytes) a **soft limit** on Java memory usage in a session, as a means to warn you if a user's session-duration Java state is using too much memory. When a user's session-duration Java state exceeds this size, Oracle generates a warning that goes into the trace files.

**See Also:**

- *Oracle Database Java Developer's Guide* for more information on this parameter
- "JAVA_MAX_SESSIONSPACE_SIZE"

# JOB_QUEUE_PROCESSES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `1000` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `0` to `1000` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`JOB_QUEUE_PROCESSES` specifies the maximum number of job slaves per instance that can be created for the execution of `DBMS_JOB` jobs and Oracle Scheduler (`DBMS_SCHEDULER`) jobs. `DBMS_JOB` and Oracle Scheduler share the same job coordinator and job slaves, and they are both controlled by the `JOB_QUEUE_PROCESSES` parameter.

If the value of `JOB_QUEUE_PROCESSES` is set to 0, then `DBMS_JOB` jobs and Oracle Scheduler jobs will not run on the instance.

If `JOB_QUEUE_PROCESSES` is set to a value in the range of `1` to `1000`, then `DBMS_JOB` jobs and Oracle Scheduler jobs will run. The actual number of job slaves created for Oracle Scheduler jobs is auto-tuned by the Scheduler depending on several factors, including available resources, Resource Manager settings, and currently running jobs. However, the combined total number of job slaves running `DBMS_JOB` jobs and Oracle Scheduler jobs on an instance can never exceed the value of `JOB_QUEUE_PROCESSES` for that instance.

Advanced replication uses Oracle Scheduler for data refreshes. Oracle Streams Advanced Queuing uses Oracle Scheduler for message propagation. Materialized views use Oracle Scheduler for automatic refreshes. Setting `JOB_QUEUE_PROCESS` to 0 will disable these features and any other features that use Oracle Scheduler or `DBMS_JOB`.

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information on the `DBMS_SCHEDULER` package
- *Oracle Database Advanced Replication* and *Oracle Database Data Warehousing Guide* for more information on managing materialized views
- *Oracle Database Advanced Queuing User's Guide* for more information about job queue processes
- *Oracle Database Administrator's Guide* for more information about the maximum number of scheduler job processes

# LARGE_POOL_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | LARGE_POOL_SIZE = *integer* [K \| M \| G] |
| **Default value** | If SGA_TARGET is set, but a value is not specified for LARGE_POOL_SIZE, then the default is 0 (internally determined by the Oracle database). If LARGE_POOL_SIZE is specified, then the user-specified value indicates a minimum value for the memory pool.<br>If SGA_TARGET is not set, 0 if both of the following are true:<br>• The pool is not required by parallel execution<br>• DBWR_IO_SLAVES is not set<br>Otherwise, derived from the values of PARALLEL_MAX_SERVERS, PARALLEL_THREADS_PER_CPU, CLUSTER_DATABASE_INSTANCES, DISPATCHERS, and DBWR_IO_SLAVES.<br>Note that a value derived in this way does not take into account the requirements used for Oracle Automatic Storage Management (Oracle ASM) files. As a general guideline, you should add 600K to the size of the SGA on a database instance using Oracle ASM. |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | Minimum: the granule size<br>Maximum: operating system-dependent |
| **Basic** | No |

LARGE_POOL_SIZE specifies (in bytes) the size of the large pool allocation heap. The large pool allocation heap is used in shared server systems for session memory, by parallel execution for message buffers, and by backup processes for disk I/O buffers. Parallel execution allocates buffers out of the large pool only when SGA_TARGET is set.

**Note:**

When Oracle derives a default value, it adds 250K for each session for the shared server if DISPATCHERS is configured. The final derived value also includes a port-specific amount of memory for backup I/O buffers. The total derived default value can either be too large to allocate or can cause performance problems. In that case, set LARGE_POOL_SIZE to a number sufficiently small so that the database can start.

You can specify the value of this parameter using a number, optionally followed by K or M to specify kilobytes or megabytes, respectively. If you do not specify K or M, then the number is taken as bytes.

**See Also:**

*Oracle Database Performance Tuning Guide* for more information on setting this parameter

# LDAP_DIRECTORY_ACCESS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `LDAP_DIRECTORY_ACCESS = { NONE \| PASSWORD \| SSL }` |
| **Default value** | `NONE` |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`LDAP_DIRECTORY_ACCESS` specifies whether Oracle refers to Oracle Internet Directory for user authentication information. If directory access is turned on, then this parameter also specifies how users are authenticated.

### Values

- `NONE`

  Oracle does not refer to Oracle Internet Directory for Enterprise User Security information.

- `PASSWORD`

  Oracle tries to connect to the enterprise directory service using the database password stored in the database wallet. If that fails, then the Oracle Internet Directory connection fails and the database will not be able to retrieve enterprise roles and schema mappings upon enterprise user login.

- `SSL`

  Oracle tries to connect to Oracle Internet Directory using SSL.

---

**See Also:**

olink:ASOAG010*Oracle Database Enterprise User Security Administrator's Guide* for more information on Enterprise User Security

---

### Using LDAP_DIRECTORY_ACCESS with PDBs

PDBs can use password and SSL authentication with Oracle Internet Directory when the default database wallet location is used.

Since there is only one `LDAP_DIRECTORY_ACCESS` initialization parameter per CDB, all the PDBs in a CDB will use the Oracle Internet Directory authentication specified by the parameter.

For a CDB, the default database wallet path is:

`ORACLE_BASE/admin/`*db-unique-name/pdb-GUID*`/wallet` (if `ORACLE_BASE` is set)

or:

`ORACLE_HOME/admin/`*db-unique-name/pdb-GUID*`/wallet` (if `ORACLE_BASE` is not set)

The exception is for the root database, which will a default wallet path of:

`ORACLE_BASE/admin/`*db-unique-name*`/wallet` (if `ORACLE_BASE` is set)

or:

`ORACLE_HOME/admin/`*db-unique-name*`/wallet` (if `ORACLE_BASE` is not set)

All the PDBs in a CDB have the same database unique name. By placing wallets in the default location, each PDB can have its own identity. Note that since there is only one `sqlnet.ora` file for a CDB, the wallet location in `sqlnet.ora` is not supported for CDBs, because each PDB must have its own wallet.

**Note:**

Oracle databases are registered with Oracle Internet Directory using Database Configuration Assistant (DBCA). For registration with Oracle Internet Directory to work, all the PDBs for a CDB must be registered using DBCA.

For non-CDBs, the default database wallet path is:

`ORACLE_BASE/admin/`*db-unique-name*`/wallet` (if `ORACLE_BASE` is set)

or:

`ORACLE_HOME/admin/`*db-unique-name*`/wallet` (if `ORACLE_BASE` is not set)

**See Also:**

*Oracle Database Enterprise User Security Administrator's Guide* for an example of setting the value of this parameter to `SSL` in the server parameter file using `ALTER SYSTEM`

# LDAP_DIRECTORY_SYSAUTH

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `LDAP_DIRECTORY_SYSAUTH = { yes \| no }` |
| **Default value** | `no` |
| **Modifiable** | No |
| **Basic** | Yes |

`LDAP_DIRECTORY_SYSAUTH` enables or disables directory-based authorization for `SYSDBA` and `SYSOPER`.

# LICENSE_MAX_SESSIONS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `1` |

| Property | Description |
|---|---|
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 0 to number of session licenses |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values, but the total for all instances mounting a database should be less than or equal to the total number of sessions licensed for that database. |

**Note:**

Oracle no longer offers licensing by the number of concurrent sessions. Therefore the `LICENSE_MAX_SESSIONS` and `LICENSE_SESSIONS_WARNING` initialization parameters have been deprecated.

`LICENSE_MAX_SESSIONS` specifies the maximum number of concurrent user sessions allowed. When this limit is reached, only users with the `RESTRICTED SESSION` privilege can connect to the database. Users who are not able to connect receive a warning message indicating that the system has reached maximum capacity.

A zero value indicates that concurrent usage (session) licensing is not enforced. If you set this parameter to a nonzero number, you might also want to set `LICENSE_SESSIONS_WARNING` (see "LICENSE_SESSIONS_WARNING").

Do not enable both concurrent usage licensing and user licensing. Set either `LICENSE_MAX_SESSIONS` or `LICENSE_MAX_USERS` to zero.

**See Also:**

*Oracle Database Administrator's Guide* for more information about this parameter

# LICENSE_MAX_USERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 0 to number of user licenses |
| **Basic** | No |
| **Oracle RAC** | Oracle recommends that multiple instances have the same value. |

`LICENSE_MAX_USERS` specifies the maximum number of users you can create in the database. When you reach this limit, you cannot create more users. You can, however, increase the limit.

**Note:**

Oracle no longer offers licensing by the number of concurrent sessions. Therefore the `LICENSE_MAX_SESSIONS` and `LICENSE_SESSIONS_WARNING` initialization parameters have been deprecated.

**See Also:**

*Oracle Database Administrator's Guide* for more information about this parameter

# LICENSE_SESSIONS_WARNING

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 0 to value of `LICENSE_MAX_SESSIONS` parameter |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

**Note:**

Oracle no longer offers licensing by the number of concurrent sessions. Therefore the `LICENSE_MAX_SESSIONS` and `LICENSE_SESSIONS_WARNING` initialization parameters have been deprecated.

`LICENSE_SESSIONS_WARNING` specifies a warning limit on the number of concurrent user sessions. When this limit is reached, additional users can connect, but Oracle writes a message in the alert log for each new connection. Users with `RESTRICTED SESSION` privilege who connect after the limit is reached receive a warning message stating that the system is nearing its maximum capacity.

If this parameter is set to zero, no warning is given as you approach the concurrent usage (session) limit. If you set this parameter to a nonzero number, you should also set `LICENSE_MAX_SESSIONS` (see "LICENSE_MAX_SESSIONS").

**See Also:**

- *Oracle Database Administrator's Guide* for more information on setting this parameter

# LISTENER_NETWORKS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | LISTENER_NETWORKS = |

```
'((NAME=network_name)
 (LOCAL_LISTENER=["]listener_address[,...]["])
 [(REMOTE_LISTENER=["]listener_address[,...]["])])'
[,...]
```

| | |
|---|---|
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

LISTENER_NETWORKS specifies one or more sets of local and remote listeners for cross-registration. All listeners within the same *network_name* will cross-register.

If a *network_name* is specified multiple times, then the resulting listener set is the union of each specification. This can be used to specify sets that require more than 255 characters, which is the per element limit.

The *listener_address* specifies a string that is an address, address list, or an alias that resolves to an address or address list of Oracle Net listeners. If an alias, the address or address list is specified in the TNSNAMES.ORA file or another address repository as configured for your system.

If a comma is used to specify a *listener_address* list, then the set of addresses must be surrounded by double quotes.

**See Also:**

*Oracle Database Net Services Administrator's Guide* for more information on setting this parameter

# LOCAL_LISTENER

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | LOCAL_LISTENER = *network_name* |
| **Default value** | (ADDRESS = (PROTOCOL=TCP)(HOST=*hostname*) (PORT=1521)) where *hostname* is the network name of the local host. |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

LOCAL_LISTENER specifies a network name that resolves to an address or address list of Oracle Net local listeners (that is, listeners that run on the same system as this

instance). The address or address list is specified in the TNSNAMES.ORA file or other address repository as configured for your system.

**See Also:**

- *Oracle Database Concepts* for more information about listener processes and dispatcher processes
- *Oracle Database Net Services Administrator's Guide* and your operating system-specific Oracle documentation for more information about specifying network addresses for the protocols on your system

# LOCK_NAME_SPACE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | LOCK_NAME_SPACE = *namespace* |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | Up to 8 alphanumeric characters. No special characters allowed. |
| **Basic** | No |

**Note:**

The LOCK_NAME_SPACE parameter is deprecated. It is retained for backward compatibility only.

LOCK_NAME_SPACE specifies the namespace that the distributed lock manager (DLM) uses to generate lock names. Consider setting this parameter if a standby or clone database has the same database name on the same cluster as the primary database.

If the standby database resides on the same file system as the primary database, then set LOCK_NAME_SPACE in the standby parameter file to a distinct value such as the following:

```
LOCK_NAME_SPACE = standby
```

# LOCK_SGA

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | No |
| **Range of values** | true \| false |

| Property | Description |
|---|---|
| **Basic** | No |

LOCK_SGA locks the entire SGA into physical memory. It is usually advisable to lock the SGA into real (physical) memory, especially if the use of virtual memory would include storing some of the SGA using disk space. This parameter is ignored on platforms that do not support it.

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about using this parameter

# LOG_ARCHIVE_CONFIG

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | LOG_ARCHIVE_CONFIG = |

```
{
[ SEND | NOSEND ]
[ RECEIVE | NORECEIVE ]
[ DG_CONFIG=(remote_db_unique_name1 [, ... remote_db_unique_name30) |
NODG_CONFIG ]
}
```

| | |
|---|---|
| **Default value** | 'SEND, RECEIVE, NODG_CONFIG' |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

LOG_ARCHIVE_CONFIG enables or disables the sending of redo logs to remote destinations and the receipt of remote redo logs, and specifies the unique database names (DB_UNIQUE_NAME) for each database in the Data Guard configuration.

### Values

- SEND

  Enables the sending of redo logs to remote destinations

- NOSEND

  Disables the sending of redo logs to remote destinations

- RECEIVE

  Enables the receipt of remotely archived redo logs

- NORECEIVE

  Disables the receipt of remotely archived redo logs

- DG_CONFIG

  Specifies a list of up to 30 unique database names (defined with the DB_UNIQUE_NAME initialization parameter) for all of the databases in the Data Guard configuration.

- NODG_CONFIG

  Eliminates the list of service provider names previously specified with the DG_CONFIG option.

**See Also:**

*Oracle Streams Concepts and Administration* for more information about setting this parameter before starting a database upgrade or maintenance operation with Oracle Streams

# LOG_ARCHIVE_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | LOG_ARCHIVE_DEST = *filespec* |
| **Default value** | Null |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | Any valid path or device name, except raw partitions |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

**Note:**

For Enterprise Edition users, this parameter has been deprecated in favor of the LOG_ARCHIVE_DEST_*n* parameters. If Oracle Enterprise Edition is not installed or it is installed, but you have not specified any LOG_ARCHIVE_DEST_*n* parameters, this parameter is valid.

LOG_ARCHIVE_DEST is applicable only if you are running the database in ARCHIVELOG mode or are recovering a database from archived redo logs. LOG_ARCHIVE_DEST is incompatible with the LOG_ARCHIVE_DEST_*n* parameters, and must be defined as the null string ("") or (' ') when any LOG_ARCHIVE_DEST_*n* parameter has a value other than a null string. Use a text string to specify the default location and root of the disk file or tape device when archiving redo log files. (Archiving to tape is not supported on all operating systems.) The value cannot be a raw partition.

If LOG_ARCHIVE_DEST is not explicitly defined and all the LOG_ARCHIVE_DEST_*n* parameters have null string values, LOG_ARCHIVE_DEST is set to an operating system-specific default value on instance startup.

To override the destination that this parameter specifies, either specify a different destination for manual archiving or use the SQL*Plus statement `ARCHIVE LOG START` *`filespec`* for automatic archiving, where *`filespec`* is the new archive destination. To permanently change the destination, use the statement `ALTER SYSTEM SET LOG_ARCHIVE_DEST =` *`filespec`*, where *`filespec`* is the new archive destination.

Neither `LOG_ARCHIVE_DEST` nor `LOG_ARCHIVE_FORMAT` have to be complete file or directory specifiers themselves; they only need to form a valid file path after the variables are substituted into `LOG_ARCHIVE_FORMAT` and the two parameters are concatenated together.

**See Also:**

- *Oracle Database Backup and Recovery User's Guide*
- "LOG_ARCHIVE_DUPLEX_DEST", "LOG_ARCHIVE_MIN_SUCCEED_DEST", and "V$ARCHIVE_DEST" for more information on setting this parameter
- Your Oracle operating system-specific documentation for the default value and for an example of how to specify the destination path or file name using `LOG_ARCHIVE_DEST`

# LOG_ARCHIVE_DEST_*n*

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `LOG_ARCHIVE_DEST_[1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 \| 10 \| 11 \| 12 \| 13 \| 14 \| 15 \| 16 \| 17 \| 18 \| 19 \| 20 \| 21 \| 22 \| 23 \| 24 \| 25 \| 26 \| 27 \| 28 \| 29 \| 30 \| 31] =` |

```
{ null_string |
{ LOCATION=path_name | SERVICE=service_name }
[ MANDATORY ]
[ REOPEN[=seconds] ]
[ DELAY[=minutes] ]
[ ENCRYPTION=ENABLED|DISABLED ]
[ NOREGISTER ]
[ TEMPLATE=template ]
[ ALTERNATE=destination ]
[ MAX_FAILURE=count ]
[ SYNC | ASYNC ]
[ AFFIRM | NOAFFIRM ]
[ NET_TIMEOUT=seconds ]
[ VALID_FOR=(redo_log_type,database_role) ]
[ DB_UNIQUE_NAME ]
[ MAX_CONNECTIONS=count ]
[ COMPRESSION={ENABLE|DISABLE} ]
}
```

| | |
|---|---|
| **Default value** | There is no default value. |

| Property | Description |
|---|---|
| **Modifiable** | Most attributes are modifiable by `ALTER SESSION` and `ALTER SYSTEM`, but some are not. See *Oracle Data Guard Concepts and Administration* for further information. |
| **Basic** | Yes |

The `LOG_ARCHIVE_DEST_`*n* initialization parameter defines up to 31 (where *n* = 1, 2, 3, ... 31) destinations, each of which *must* specify either the `LOCATION` or the `SERVICE` attribute to specify where to archive the redo data. All other attributes are optional. Note that whether you are specifying the `LOCATION` attribute or the `SERVICE` attribute, it must be the first attribute supplied in the list of attributes.

If you choose not to enter any attributes, then you can specify a NULL string by entering the following:

```
LOG_ARCHIVE_DEST_n=' ';
```

You set the attributes for the `LOG_ARCHIVE_DEST_`*n* initialization parameter to control different aspects of how redo transport services transfer redo data from a production or primary database destination to another (standby) database destination. You can query the `V$ARCHIVE_DEST` view to see the current attribute settings for each destination (`n`).

**Note:**

Some of the attributes for this parameter are deprecated, but are being retained for backward compatibility. See "Deprecated Attributes for LOG_ARCHIVE_DEST_n".

For every `LOG_ARCHIVE_DEST_n` initialization parameter that you define, you must specify a corresponding `LOG_ARCHIVE_DEST_STATE_n` parameter. The `LOG_ARCHIVE_DEST_STATE_n` (where `n` is an integer from 1 to 31) initialization parameter specifies whether the corresponding destination is currently enabled or disabled.

**See Also:**

"LOG_ARCHIVE_DEST_STATE_n"

Destinations `LOG_ARCHIVE_DEST_11` through `LOG_ARCHIVE_DEST_31` do not support the `SYNC`, `ARCH`, `LOCATION`, `MANDATORY`, or `ALTERNATE` attributes, and cannot be specified as the target of the `ALTERNATE` attribute.

`LOG_ARCHIVE_DEST_11` through `LOG_ARCHIVE_DEST_31` can only be used when the `COMPATIBLE` initialization parameter is set to `11.2.0` or higher.

### Values

**See Also:**

*Oracle Data Guard Concepts and Administration* for detailed descriptions of all the values listed in this section

- `AFFIRM` and `NOAFFIRM`

  Control whether a redo transport destination acknowledges received redo data before or after writing it to the standby redo log. The default is `NOAFFIRM`.

- `ALTERNATE=LOG_ARCHIVE_DEST_`*n*

  Specifies an alternate archiving destination to be used when the original destination fails. There is no default value; if an alternate destination is not specified, then archiving does not automatically change to another destination if the original destination fails.

- `ASYNC`

  The redo data generated by a transaction need not have been received at a destination which has this attribute before that transaction can commit. This is the default behavior if neither `SYNC` nor `ASYNC` is specified.

- `COMPRESSION`

  Indicates whether network compression is enabled or disabled.

  The `COMPRESSION` attribute is ignored if the Advanced Compression option has not been enabled.

- `DB_UNIQUE_NAME=`*name*

  Specifies a unique name for the database at this destination. You must specify a name; there is no default value.

- `DELAY`[=*minutes*]

  Specifies a minimum time lag between when redo data is archived on a standby site retrieving redo from a primary and when the archived redo log file is applied to the standby database or any standbys cascading from it. If you specify the DELAY attribute without a time interval, the default is 30 minutes.

- `ENCRYPTION=DISABLE|ENABLE`

  Controls whether encryption of the redo stream sent to Zero Data Loss Recovery Appliance (Recovery Appliance) is enabled or disabled. The default value for the attribute is `DISABLE`.

  This attribute cannot be used with the `COMPRESSION`, `LOCATION`, or `SYNC` attributes.

  To use the `ENCRYPTION` attribute, you must set the `COMPATIBLE` initialization parameter to `11.2.0.4` or higher on the protected database.

**See Also:**

*Zero Data Loss Recovery Appliance Administrator's Guide* for introductory information about Recovery Appliance

- `LOCATION`=*local_disk_directory* or `USE_DB_RECOVERY_FILE_DEST`

  Specifies either a local file system destination or the directory, file system, or Automatic Storage Management disk group that will serve as the fast recovery area. You must specify this attribute for at least one destination. You can specify either a local disk directory or fast recovery area with the `LOCATION` attribute. You *must* include either the `LOCATION` or the `SERVICE` attribute for each destination to specify where to archive the redo data.

- `MANDATORY`

  Specifies that the transmission of redo data to the destination must succeed before the local online redo log file can be made available for reuse. If the `MANDATORY` attribute is not specified, then the destination is optional.

- `MAX_CONNECTIONS`

  Specifies the maximum number of network connections that can be used to transmit redo data to this destination. The default is 1.

- `MAX_FAILURE`

  Controls the consecutive number of times redo transport services attempt to reestablish communication and transmit redo data to a failed destination before the primary database permanently gives up on the standby database.

- `NET_TIMEOUT`=*seconds*

  Specifies the number of seconds the log writer process on the primary system waits for status from the SYNC (NSS*n*) process before terminating the network connection. The default is 30 seconds.

- `NOREGISTER`

  Indicates that the location of the archived redo log file is not to be recorded at the corresponding destination.

- `REOPEN`[=*seconds*]

  Specifies the minimum number of seconds before the archiver processes (ARC*n*) or the log writer process (LGWR) should try again to access a previously failed destination. The default is 300 seconds.

- `SERVICE`=*net_service_name*

  Specifies a valid Oracle Net service name (`SERVICE=`*net_service_name*) that identifies the remote Oracle database instance to which redo data will be sent. Each destination *must* specify either the `LOCATION` or the `SERVICE` attribute. There is no default net service name.

- `SYNC`

  The redo data generated by a transaction must have been received by every enabled destination which has this attribute before that transaction can commit.

- `TEMPLATE`=*filename_template*_%t_%s_%r

Specifies a path name and a file name template for archived redo log files created at a redo transport destination that contain redo data from the database where this attribute is specified. This attribute overrides the value of the LOG_ARCHIVE_FORMAT initialization parameter at a redo transport destination. This attribute does not have a default value.

- VALID_FOR=(redo_log_type, database_role)

  Identifies when redo transport services can transmit redo data to destinations based on the following factors:

  – redo_log_type—whether online redo log files, standby redo log files, or both are *currently* being archived on the database at this destination

  – database_role—whether the database is *currently* running in the primary or the standby role

### Deprecated Attributes for LOG_ARCHIVE_DEST_n

The following attributes are deprecated for the LOG_ARCHIVE_DEST_*n* parameter. They are retained for backward compatibility only.

***Table 1-3 Deprecated Attributes on the LOG_ARCHIVE_DEST_n Initialization Parameter***

| Deprecated Attribute | Alternative |
|---|---|
| ARCH | Specify SYNC or ASYNC. ASYNC is the default if neither attribute is specified. |
| LGWR | Specify SYNC or ASYNC. ASYNC is the default if neither attribute is specified. |
| OPTIONAL | Destinations are optional by default. |
| VERIFY | None. This attribute is only used with the deprecated ARCH attribute. |

In addition, note the following changes to the ASYNC and SYNC attributes:

- The BLOCKS keyword on the ASYNC attribute is no longer needed.

  It is no longer necessary to set this keyword because Data Guard dynamically adjusts the block count up or down to an appropriate number of blocks, as necessary.

- The PARALLEL and NOPARALLEL keywords on the SYNC attribute are no longer needed.

## LOG_ARCHIVE_DEST_STATE_*n*

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | LOG_ARCHIVE_DEST_STATE_[1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 \| 10 \| 11 \| 12 \| 13 \| 14 \| 15 \| 16 \| 17 \| 18 \| 19 \| 20 \| 21 \| 22 \| 23 \| 24 \| 25 \| 26 \| 27 \| 28 \| 29 \| 30 \| 31] = { enable \| defer \| alternate } |

| Property | Description |
|---|---|
| **Default value** | `enable` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | Yes |

The `LOG_ARCHIVE_DEST_STATE_`*n* parameters (where *n* = 1, 2, 3, ... 31) specify the availability state of the corresponding destination. The parameter suffix (1 through 31) specifies one of the corresponding `LOG_ARCHIVE_DEST_`*n* destination parameters.

### Values

- `enable`

  Specifies that a valid log archive destination can be used for a subsequent archiving operation (automatic or manual). This is the default.

- `defer`

  Specifies that valid destination information and attributes are preserved, but the destination is excluded from archiving operations until reenabled.

- `alternate`

  Specifies that a log archive destination is not enabled but will become enabled if communications to another destination fail.

The `LOG_ARCHIVE_DEST_STATE_`*n* parameters have no effect on the `ENABLE` state for the `LOG_ARCHIVE_DEST` or `LOG_ARCHIVE_DUPLEX_DEST` parameters.

The `V$ARCHIVE_DEST` dynamic performance view shows values in use for the current session. The `DEST_ID` column of that view corresponds to the archive destination suffix *n*.

**See Also:**

- *Oracle Data Guard Concepts and Administration* for more information about this parameter
- "V$ARCHIVE_DEST"

# LOG_ARCHIVE_DUPLEX_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `LOG_ARCHIVE_DUPLEX_DEST = `*`filespec`* |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Either a null string or any valid path or device name, except raw partitions |

| Property | Description |
|---|---|
| **Basic** | No |

**Note:**

If you are using Oracle Enterprise Edition, this parameter is deprecated in favor of the LOG_ARCHIVE_DEST_*n* parameters. If Oracle Enterprise Edition is not installed or it is installed but you have not specified any LOG_ARCHIVE_DEST_*n* parameters, this parameter is valid.

LOG_ARCHIVE_DUPLEX_DEST is similar to the initialization parameter LOG_ARCHIVE_DEST. This parameter specifies a second archive destination: the **duplex** archive destination. This duplex archive destination can be either a must-succeed or a best-effort archive destination, depending on how many archive destinations must succeed (as specified in the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter).

The default setting of a null string ("") or (' ') indicates that a duplex archive destination does not exist.

**See Also:**

- "LOG_ARCHIVE_DEST_n"
- "LOG_ARCHIVE_MIN_SUCCEED_DEST"
- "V$ARCHIVE_DEST"
- *Oracle Database Administrator's Guide* for an example of using this parameter to specify an optional secondary archive destination

# LOG_ARCHIVE_FORMAT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | LOG_ARCHIVE_FORMAT = *filename* |
| **Default value** | Operating system-dependent |
| **Modifiable** | No |
| **Range of values** | Any string that resolves to a valid filename |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values, but identical values are recommended. |

LOG_ARCHIVE_FORMAT is applicable only if you are using the redo log in ARCHIVELOG mode. Use a text string and variables to specify the default filename format when archiving redo log files. The string generated from this format is appended to the string specified in the LOG_ARCHIVE_DEST parameter.

The following variables can be used in the format:

**%s** log sequence number

**%S** log sequence number, zero filled

**%t** thread number

**%T** thread number, zero filled

**%a** activation ID

**%d** database ID

**%r** resetlogs ID that ensures unique names are constructed for the archived log files across multiple incarnations of the database

Using uppercase letters for the variables (for example, %S) causes the value to be fixed length and padded to the left with zeros. An example of specifying the archive redo log filename format follows:

```
LOG_ARCHIVE_FORMAT = 'log%t_%s_%r.arc'
```

Archive log file names must contain each of the elements %s (sequence), %t (thread), and %r (resetlogs ID) to ensure that all archive log file names are unique. If the LOG_ARCHIVE_FORMAT initialization parameter is set in the parameter file, then make sure the parameter value contains the %s, %t, and %r elements. Otherwise, the following error is displayed at the time of instance startup:

```
ORA-19905: log_archive_format must contain %s, %t and %r
```

Neither LOG_ARCHIVE_DEST nor LOG_ARCHIVE_FORMAT have to be complete file or directory specifiers themselves; they only need to form a valid file path after the variables are substituted into LOG_ARCHIVE_FORMAT and the two parameters are concatenated together.

LOG_ARCHIVE_FORMAT is ignored in these cases:

- For archived log files that go to the fast recovery area
- When LOG_ARCHIVE_DEST[_n] points to the root of an Oracle ASM disk group (for example, +DATA). The directory of a disk group (for example, +DATA/logs) must be specified for the parameter to be honored.

In these cases where LOG_ARCHIVE_FORMAT is ignored, an Oracle ASM file name is used. See *Oracle Automatic Storage Management Administrator's Guide* for more information on Oracle ASM file names.

**See Also:**

- *Oracle Database Backup and Recovery User's Guide*, *Oracle Data Guard Concepts and Administration*, and *Oracle Real Application Clusters Administration and Deployment Guide* for more information about this parameter
- Your operating system- specific Oracle documentation for the default value and range of values for LOG_ARCHIVE_FORMAT

# LOG_ARCHIVE_MAX_PROCESSES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 4 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 1 to 30 |
| **Basic** | No |

LOG_ARCHIVE_MAX_PROCESSES specifies the maximum number of ARC*n* processes that can be created.

**See Also:**

*Oracle Database Administrator's Guide* for more information about LOG_ARCHIVE_MAX_PROCESSES

# LOG_ARCHIVE_MIN_SUCCEED_DEST

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 1 |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | 1 to 10 if you are using LOG_ARCHIVE_DEST_*n*<br>1 or 2 if you are using LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST |
| **Basic** | No |

LOG_ARCHIVE_MIN_SUCCEED_DEST defines the minimum number of destinations that must succeed in order for the online logfile to be available for reuse.

- If you are using the LOG_ARCHIVE_DEST_*n* parameters and automatic archiving is enabled, then the value of this parameter cannot exceed the number of enabled, valid destinations specified as MANDATORY plus the number of enabled, valid destinations that are configured with the OPTIONAL and LOCATION attributes.
- If you are using LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST and automatic archiving is enabled, a value of 1 specifies that the destination specified in LOG_ARCHIVE_DEST must succeed. A value of 2 specifies that the destinations specified in both parameters must succeed.

If the value of this parameter is less than the number of enabled, valid MANDATORY destinations, this parameter is ignored in favor of the MANDATORY destination count. If the value is more than the number of enabled, valid MANDATORY destinations, then

some of the enabled, valid destinations configured with the OPTIONAL and LOCATION attributes are treated as MANDATORY.

You can switch dynamically from using the older parameters to the LOG_ARCHIVE_DEST_*n* parameter using ALTER SYSTEM, as follows:

1. Set LOG_ARCHIVE_MIN_SUCCEED_DEST to 1.
2. Set the value of LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST to the null string.
3. Set the desired number of destinations for the LOG_ARCHIVE_DEST_*n* parameters.
4. Reset LOG_ARCHIVE_MIN_SUCCEED_DEST to the desired value.

**See Also:**

- *Oracle Database Administrator's Guide* for more information on setting this parameter
- "LOG_ARCHIVE_DEST_n", "LOG_ARCHIVE_DUPLEX_DEST", and "V$ARCHIVE_DEST" for information on related parameters

# LOG_ARCHIVE_TRACE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768 |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

The LOG_ARCHIVE_TRACE parameter enables and controls the generation of comprehensive trace information for log archiving and redo transport activity. The additional tracing that is output when setting LOG_ARCHIVE_TRACE to a non-zero value can appear in trace files for an archive process, RFS process, LGWR process, SYNC process, ASYNC process, foreground process, MRP process, recovery process, log apply process, startup process, shutdown process, and other processes that use redo transport services.

Table 1-4 describes the valid LOG_ARCHIVE_TRACE values.

***Table 1-4 LOG_ARCHIVE_TRACE Values***

| Value | Description | More Information |
|---|---|---|
| 0 | Disables archivelog tracing (this is the default) | |

| Value | Description | More Information |
|---|---|---|
| 1 | High-level tracing | High-level tracing provides a small amount of tracing from most redo transport processes. When a value is specified that is not high-level (not 1), the high-level tracing for that value will be output to the trace file even if the high-level tracing has not been specifically requested. For example, if you set `LOG_ARCHIVE_TRACE` to 128, you will get all the trace messages pertaining to the FAL engine, *including* the high-level tracing from the FAL engine. |
| 2 | Tracks Data Guard interfaces | This tracing level is for the various Data Guard configuration parameters, the various Data Guard related SQL commands, the various Data Guard table accesses, and some Data Guard internal interfaces. |
| 4 | Tracks common redo transport services | This tracing level is for a number of common services provided by redo transport to various internal users (such as the RFS process, the LGWR process, and the ASYNC process). These include locking primitives, use of common VOS (Virtual Operating System) interfaces, task scheduling, process monitoring, control file access, and other miscellaneous common services. |
| 8 | Tracks database protection mode | |
| 16 | Tracks disk and network I/O requests | |
| 32 | Tracks redo transport destinations | |
| 64 | Tracks redo transport archive engine | |
| 128 | Tracks redo transport FAL (fetch archive log) engine | |
| 256 | Tracks Physical, Logical, and Availability Machine clients of RFS process | |
| 512 | Tracks redo transport of LGWR, SYNC, and ASYNC processes | |
| 1024 | Tracks redo transport RFS process | |
| 2048 | Tracks redo transport gap resolution | |
| 4096 | Tracks real-time apply | |
| 8192 | Tracks recovery process | |
| 16384 | Tracks redo transport buffer management | |

| Value | Description | More Information |
|---|---|---|
| 32768 | Tracks LogMiner dictionary | |

You can combine tracing levels by adding together the values of the desired tracing levels. For example, a setting of 3 will generate level 1 and level 2 trace output. You can set different values for the primary and standby database.

When this parameter is set to the default value of 0, Oracle will still generate appropriate alert and trace entries in response to error conditions. If you change the value of this parameter dynamically in an `ALTER SYSTEM` statement, the change takes effect immediately.

Many of the trace messages from redo transport processes include a routine name at the beginning of the trace message. The first part of the routine name specifies the module for that routine. Table 1-5 shows the tracing value used for each redo transport module. Note that the tracing value for transport modules can change from release to release.

***Table 1-5 Tracing Values Used for Redo Transport Modules***

| Module Name | Purpose | Tracing Value |
|---|---|---|
| krsa | Locking primitives | 4 |
| krsb | Buffer management | 16384 |
| krsc | Crash recovery | 64 |
| krsd | Destination | 32 |
| krse | Archive engine | 64 |
| krsf | FAL engine | 128 |
| krsg | Gap resolution | 2048 |
| krsh | Helper | 4 |
| krsi | Input/output | 16 |
| krsj | Protection mode | 8 |
| krsk | Control file access | 4 |
| krsl | Log writer | 512 |
| krsm | MRP process | 8192 |
| krsn | Configuration | 2 |
| krso | Process monitor | 4 |
| krsp | Physical standby and Availability Machine RFS client | 256 |
| krsq | SQL commands | 2 |
| krsr | RFS process | 1024 |
| krst | Tables | 2 |

| Module Name | Purpose | Tracing Value |
|---|---|---|
| krsu | Network I/O | 16 |
| krsv | VOS front end | 4 |
| krsw | ASYNC process | 512 |
| krsx | Internal interfaces | 2 |

**See Also:**

*Oracle Database Administrator's Guide*

# LOG_BUFFER

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Default value** | 5 MB to 32 MB, depending on the size of the SGA, CPU count, and whether the operating system is 32-bit or 64-bit |
| **Modifiable** | No |
| **Range of values** | 2 MB to operating system-dependent |
| **Basic** | No |

`LOG_BUFFER` specifies the amount of memory (in bytes) that Oracle uses when buffering redo entries to a redo log file. Redo log entries contain a record of the changes that have been made to the database block buffers. The LGWR process writes redo log entries from the log buffer to a redo log file.

The log buffer size depends on the number of redo strands in the system. One redo strand is allocated for every 16 CPUs and has a default size of 2 MB. Oracle allocates a minimum of 2 redo strands per instance. When the log buffer size is not specified, any remaining memory in the redo granules is given to the log buffer.

**See Also:**

- Your operating system-specific Oracle documentation for the default value and range of values
- *Oracle Streams Replication Administrator's Guide* for information on using this parameter in an Oracle Streams environment
- *Oracle Database Performance Tuning Guide* for information on resizing the redo log buffer using this parameter

# LOG_CHECKPOINT_INTERVAL

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `0` to $2^{31}$ - 1 |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`LOG_CHECKPOINT_INTERVAL` specifies the frequency of checkpoints in terms of the number of redo log file blocks that can exist between an incremental checkpoint and the last block written to the redo log. This number refers to physical operating system blocks, not database blocks.

Regardless of this value, a checkpoint always occurs when switching from one online redo log file to another. Therefore, if the value exceeds the actual redo log file size, checkpoints occur only when switching logs. Checkpoint frequency is one of the factors that influence the time required for the database to recover from an unexpected failure.

**Note:**

- Specifying a value of `0` (zero) for `LOG_CHECKPOINT_INTERVAL` has the same effect as setting the parameter to infinity and causes the parameter to be ignored. Only nonzero values of this parameter are considered meaningful.
- Recovery I/O can also be limited by setting the `LOG_CHECKPOINT_TIMEOUT` parameter or by the size specified for the smallest redo log. For information on which mechanism is controlling checkpointing behavior, query the `V$INSTANCE_RECOVERY` view.

**See Also:**

- "LOG_CHECKPOINT_TIMEOUT"
- "V$INSTANCE_RECOVERY"
- *Oracle Database Performance Tuning Guide* for information on disabling or removing this parameter when the `FAST_START_MTTR_TARGET` parameter is set

# LOG_CHECKPOINT_TIMEOUT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `0` to $2^{31}$ - 1 |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`LOG_CHECKPOINT_TIMEOUT` specifies (in seconds) the amount of time that has passed since the incremental checkpoint at the position where the last write to the redo log (sometimes called the **tail of the log**) occurred. This parameter also signifies that no buffer will remain dirty (in the cache) for more than *integer* seconds.

Specifying a value of `0` for the timeout disables time-based checkpoints. Hence, setting the value to `0` is not recommended unless `FAST_START_MTTR_TARGET` is set.

**Note:**

- A checkpoint scheduled to occur because of this parameter is delayed until the completion of the previous checkpoint if the previous checkpoint has not yet completed.
- Recovery I/O can also be limited by setting the `LOG_CHECKPOINT_INTERVAL` parameter or by the size specified for the smallest redo log. For information on which mechanism is controlling checkpointing behavior, query the `V$INSTANCE_RECOVERY` view.

**See Also:**

- "LOG_CHECKPOINT_TIMEOUT"
- "V$INSTANCE_RECOVERY"
- *Oracle Database Performance Tuning Guide* for information on disabling or removing this parameter when the `FAST_START_MTTR_TARGET` parameter is set

# LOG_CHECKPOINTS_TO_ALERT

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |

| Property | Description |
|---|---|
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`LOG_CHECKPOINTS_TO_ALERT` lets you log your checkpoints to the alert log. Doing so is useful for determining whether checkpoints are occurring at the desired frequency.

# LOG_FILE_NAME_CONVERT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `LOG_FILE_NAME_CONVERT = '`*`string1`*`' , '`*`string2`*`' , '`*`string3`*`' , '`*`string4`*`' , ...`<br>Where:<br>• *`string1`* is the pattern of the primary database filename<br>• *`string2`* is the pattern of the standby database filename<br>• *`string3`* is the pattern of the primary database filename<br>• *`string4`* is the pattern of the standby database filename<br>You can use as many pairs of primary and standby replacement strings as required. You can use single or double quotation marks.<br>The following are example settings that are acceptable:<br>`LOG_FILE_NAME_CONVERT = '/dbs/t1/','/dbs/t1/s','dbs/t2/ ','dbs/t2/s_'` |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Basic** | No |

`LOG_FILE_NAME_CONVERT` converts the filename of a new log file on the primary database to the filename of a log file on the standby database. If you add a log file to the primary database, you must add a corresponding file to the standby database.

If you specify an odd number of strings (the last string has no corresponding replacement string), an error is signalled during startup. If the filename being converted matches more than one pattern in the pattern/replace string list, the first matched pattern takes effect. There is no limit on the number of pairs that you can specify in this parameter (other than the hard limit of the maximum length of multivalue parameters).

When the standby database is updated, this parameter converts the log file name on the primary database to the log file name on the standby database. The file must exist on the standby database and must be writable or the recovery process will halt with an error.

The first string is the pattern found in the log file names on the primary database. The second string is the pattern found in the log file names on the standby database.

You should also use LOG_FILE_NAME_CONVERT to rename the logfiles in the clone control file when setting up the clone database during tablespace point-in-time recovery.

**Note:**

The LOG_FILE_NAME_CONVERT parameter applies only to online logs (not to archived logs).

**See Also:**

*Oracle Database Backup and Recovery User's Guide* and *Oracle Data Guard Concepts and Administration*

# MAX_DISPATCHERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | If MAX_DISPATCHERS is specified, then it should be greater than or equal to the number of dispatchers specified by the DISPATCHERS parameter and less than the number of processes specified by the PROCESSES parameter. |
| **Basic** | No |

MAX_DISPATCHERS specifies the maximum number of dispatcher processes allowed to be running simultaneously. It can be overridden by the DISPATCHERS parameter and is maintained for backward compatibility with older releases.

**See Also:**

- *Oracle Database Administrator's Guide* for more information about this parameter
- Your operating system-specific Oracle documentation for the default value and range of values

# MAX_DUMP_FILE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | MAX_DUMP_FILE_SIZE = { *integer* [K \| M \| G] \| UNLIMITED } |
| **Default value** | UNLIMITED |

| Property | Description |
|---|---|
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | 0 to unlimited, or `UNLIMITED` |
| **Basic** | No |

`MAX_DUMP_FILE_SIZE` specifies the maximum size of trace files (excluding the alert log). Change this limit if you are concerned that trace files may use too much space.

- A numeric value for `MAX_DUMP_FILE_SIZE` specifies the maximum size in operating system blocks.
- A numeric value followed by a `K` or `M` or `G` suffix specifies the file size in kilobytes, megabytes, or gigabytes.
- The special value string `UNLIMITED` means that there is no upper limit on trace file size. Thus, dump files can be as large as the operating system permits.

When the trace file is limited in size, it may be automatically split into multiple files, called segments, if needed. The segments will have the same file name as the active trace file, but with an extra segment number appended.

The trace file can be split into a maximum of 5 segments, and the size of each segment will typically be 1/5th of the trace file limit.

When the combined size of all the trace file segments exceeds the specified limit, the oldest segment is deleted, and a new, empty segment is created. Thus, the trace file always contains the most recent trace information.

Note that the first segment is never deleted, because it may contain relevant information about the initial state of the process.

**See Also:**

*Oracle Database Administrator's Guide* and *Oracle Database SQL Tuning Guide* for more information on setting this parameter

# MAX_ENABLED_ROLES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `30` |
| **Modifiable** | No |
| **Range of values** | 1 to 148 |
| **Basic** | No |

**Note:**

The MAX_ENABLED_ROLES parameter is deprecated. It is retained for backward compatibility only.

Even if valid values are assigned to this parameter, they are ignored. There is no need to explicitly disable the parameter.

MAX_ENABLED_ROLES specifies the maximum number of database roles that users can enable, including roles contained within other roles. The actual number of roles that users can enable is 2 plus the value of MAX_ENABLED_ROLES, because each user has two additional roles, PUBLIC and the user's own role.

For example, if MAX_ENABLED_ROLES is set to 5, then user HR can have seven roles enabled (the five enabled by MAX_ENABLED_ROLES, plus PUBLIC and HR).

# MAX_SHARED_SERVERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | If MAX_SHARED_SERVERS is specified, then it should be greater than or equal to SHARED_SERVERS and less than PROCESSES. |
| **Basic** | No |

MAX_SHARED_SERVERS specifies the maximum number of shared server processes allowed to be running simultaneously. Setting this parameter enables you to reserve process slots for other processes, such as dedicated servers.

When you want to reduce the range of shared servers, you can reduce MAX_SHARED_SERVERS before reducing SHARED_SERVERS. If MAX_SHARED_SERVERS is lower than SHARED_SERVERS, then the number of shared servers will not vary but will remain at the constant level specified by SHARED_SERVERS. If MAX_SHARED_SERVERS is not specified, then a shared server process may be spawned as long as the number of free process slots is greater than 1 / 8 the maximum number of processes, or 2 if PROCESSES is less than 24.

**See Also:**

- "SHARED_SERVERS"
- "PROCESSES"
- *Oracle Database Administrator's Guide* for more information on setting this parameter
- *Oracle Database Concepts* for information on processes
- Your operating system-specific Oracle documentation for the default value and range of values

# MAX_STRING_SIZE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `MAX_STRING_SIZE = { STANDARD \| EXTENDED }` |
| **Default value** | `STANDARD` |
| **Modifiable** | `ALTER SYSTEM` ... SID='*'[1] |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must use the same value. |

[1] Use ALTER SYSTEM only when the database is in UPGRADE mode, and run the utl32k.sql script afterward, as explained in this section.

`MAX_STRING_SIZE` controls the maximum size of `VARCHAR2`, `NVARCHAR2`, and `RAW` data types in SQL.

`STANDARD` means that the length limits for Oracle Database releases prior to Oracle Database 12*c* apply (for example, 4000 bytes for `VARCHAR2` and `NVARCHAR2`, and 2000 bytes for `RAW`).

`EXTENDED` means that the 32767 byte limit introduced in Oracle Database 12*c* applies.

The `COMPATIBLE` initialization parameter must be set to `12.0.0.0` or higher to set `MAX_STRING_SIZE` = `EXTENDED`.

You can change the value of `MAX_STRING_SIZE` from `STANDARD` to `EXTENDED`. However, you cannot change the value of `MAX_STRING_SIZE` from `EXTENDED` to `STANDARD`.

By setting `MAX_STRING_SIZE` = `EXTENDED`, users are taking an explicit action that could introduce application incompatibility in their database. Applications that do not want to use the expanded data types can be rewritten for compatibility with either setting; for example, these applications could use explicit CASTs to fix the length of `VARCHAR2` expressions during `CREATE TABLE AS SELECT`.

Altering `MAX_STRING_SIZE` will update database objects and possibly invalidate them, as follows:

- Tables with virtual columns will be updated with new data type metadata for virtual columns of `VARCHAR2(4000)`, 4000-byte `NVARCHAR2`, or `RAW(2000)` type.
  - Functional indexes will become unusable if a change to their associated virtual columns causes the index key to exceed index key length limits. Attempts to rebuild such indexes will fail with `ORA-01450: maximum key length exceeded`.
- Views will be invalidated if they contain `VARCHAR2(4000)`, 4000-byte `NVARCHAR2`, or `RAW(2000)` typed expression columns.
- Materialized views will be updated with new metadata `VARCHAR2(4000)`, 4000-byte `NVARCHAR2`, and `RAW(2000)` typed expression columns

### Increasing the Maximum Size of VARCHAR2, NVARCHAR2, and RAW Columns in a Non-CDB

To increase the maximum size of VARCHAR2, NVARCHAR2, and RAW columns in a non-CDB:

1. Shut down the database.
2. Restart the database in UPGRADE mode.
3. Change the setting of MAX_STRING_SIZE to EXTENDED.
4. Run the rdbms/admin/utl32k.sql script. You must be connected AS SYSDBA to run the script.
5. Restart the database in NORMAL mode.

---

**Note:**

The utl32k.sql script increases the maximum size of the VARCHAR2, NVARCHAR2, and RAW columns for the views where this is required. The script does not increase the maximum size of the VARCHAR2, NVARCHAR2, and RAW columns in some views because of the way the SQL for those views is written.

---

6. Run the rdbms/admin/utlrp.sql script to recompile invalid objects. You must be connected AS SYSDBA to run the script.

### Increasing the Maximum Size of VARCHAR2, NVARCHAR2, and RAW Columns in a PDB

To increase the maximum size of VARCHAR2, NVARCHAR2, and RAW columns in a PDB:

1. Shut down the PDB.
2. Reopen the PDB in migrate mode.

---

**Note:**

The following SQL statement can be used to reopen a PDB in migrate mode when the current container is the PDB:

```
ALTER PLUGGABLE DATABASE pdb-name OPEN UPGRADE;
```

---

3. Change the setting of MAX_STRING_SIZE in the PDB to EXTENDED.
4. Run the rdbms/admin/utl32k.sql script in the PDB. You must be connected AS SYSDBA to run the utl32k.sql script.
5. Reopen the PDB in NORMAL mode.

---

**Note:**

The utl32k.sql script increases the maximum size of the VARCHAR2, NVARCHAR2, and RAW columns for the views where this is required. The script does not increase the maximum size of the VARCHAR2, NVARCHAR2, and RAW columns in some views because of the way the SQL for those views is written.

---

6. Run the `rdbms/admin/utlrp.sql` script in the PDB to recompile invalid objects. You must be connected `AS SYSDBA` to run the script.

---

**See Also:**

*Oracle Database Administrator's Guide* for more information about modifying the open mode of PDBs.

---

### Increasing the Maximum Size of VARCHAR2, NVARCHAR2, and RAW Columns in an Oracle RAC Database

To increase the maximum size of `VARCHAR2`, `NVARCHAR2`, and `RAW` columns in an Oracle RAC database:

1. Shut down all of the Oracle RAC database instances, except one.
2. Restart the Oracle RAC database instance in `UPGRADE` mode.
3. Change the setting of `MAX_STRING_SIZE` to `EXTENDED`.
4. Run the `rdbms/admin/utl32k.sql` script in the Oracle RAC database instance. You must be connected `AS SYSDBA` to run the script.
5. Restart all Oracle RAC database instances in `NORMAL` mode.

---

**Note:**

The `utl32k.sql` script increases the maximum size of the `VARCHAR2`, `NVARCHAR2`, and `RAW` columns for the views where this is required. The script does not increase the maximum size of the `VARCHAR2`, `NVARCHAR2`, and `RAW` columns in some views because of the way the SQL for those views is written.

---

6. Run the `rdbms/admin/utlrp.sql` script to recompile invalid objects. You must be connected `AS SYSDBA` to run the script.

### Increasing the Maximum Size of VARCHAR2, NVARCHAR2, and RAW Columns in an Oracle Data Guard Logical Standby Database

To increase the maximum size of `VARCHAR2`, `NVARCHAR2`, and `RAW` columns in an Oracle Data Guard logical standby database:

1. Shut down the Oracle Data Guard primary database and logical standby database.
2. Restart the primary database and logical standby database in `UPGRADE` mode.
3. Change the setting of `MAX_STRING_SIZE` to `EXTENDED` on the primary database and logical standby database.
4. Run the `rdbms/admin/utl32k.sql` script on both the primary database and the logical standby database. You must be connected `AS SYSDBA` to run the script.
5. Restart the primary database and logical standby database in `NORMAL` mode.

**Note:**

The `utl32k.sql` script increases the maximum size of the `VARCHAR2`, `NVARCHAR2`, and `RAW` columns for the views where this is required. The script does not increase the maximum size of the `VARCHAR2`, `NVARCHAR2`, and `RAW` columns in some views because of the way the SQL for those views is written.

6. Run the `rdbms/admin/utlrp.sql` script on the primary database and logical standby database to recompile invalid objects. You must be connected `AS SYSDBA` to run the script.
7. Restart SQL Apply.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about the `MAX_STRING_SIZE` parameter

# MEMORY_MAX_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `MEMORY_MAX_TARGET = integer [K \| M \| G]` |
| **Default value** | `0` |
| **Modifiable** | No |
| **Range of values** | `0` to the physical memory size available to the Oracle Database |
| **Basic** | No |

`MEMORY_MAX_TARGET` specifies the maximum value to which a DBA can set the `MEMORY_TARGET` initialization parameter. See the description of `MEMORY_TARGET` for more information about how the settings of `MEMORY_MAX_TARGET` and `MEMORY_TARGET` affect each other.

**See Also:**

*Oracle Database Administrator's Guide* for more information about managing memory

# MEMORY_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `MEMORY_TARGET = integer [K \| M \| G]` |
| **Default value** | `0` (SGA autotuning is disabled for `DEFERRED` mode autotuning requests, but allowed for `IMMEDIATE` mode autotuning requests) |

| Property | Description |
|---|---|
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 152 MB to MEMORY_MAX_TARGET |
| **Basic** | No |

MEMORY_TARGET specifies the Oracle systemwide usable memory. The database tunes memory to the MEMORY_TARGET value, reducing or enlarging the SGA and PGA as needed.

MEMORY_TARGET should be set higher than or equal to the sum of the current sizes of the SGA and PGA.

In a text-based initialization parameter file, if you omit MEMORY_MAX_TARGET and include a value for MEMORY_TARGET, then the database automatically sets MEMORY_MAX_TARGET to the value of MEMORY_TARGET. If you omit the line for MEMORY_TARGET and include a value for MEMORY_MAX_TARGET, the MEMORY_TARGET parameter defaults to zero. After startup, you can then dynamically change MEMORY_TARGET to a nonzero value, provided that it does not exceed the value of MEMORY_MAX_TARGET.

Total memory usage can grow beyond the value of MEMORY_TARGET. For example, memory is allocated to PL/SQL tables and varrays regardless of the value of MEMORY_TARGET as long as memory is available at the operating system level.

In the **Default value** field, IMMEDIATE mode autotuning requests are necessary to avoid ORA-04031 errors. The DEFERRED and IMMEDIATE modes are reflected in the OPER_MODE column of the V$MEMORY_RESIZE_OPS view.

**Note:**

The default value of SGA_MAX_SIZE depends on the values of MEMORY_TARGET and MEMORY_MAX_TARGET.

**See Also:**

*Oracle Database Administrator's Guide* for more information about managing memory

# NLS_CALENDAR

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_CALENDAR = "*calendar_system*" |
| **Default value** | None, implies GREGORIAN |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid calendar format name |
| **Basic** | No |

NLS_CALENDAR specifies which calendar system Oracle uses. It can have one of the following values:

- Arabic Hijrah
- English Hijrah
- Gregorian
- Japanese Imperial
- Persian
- ROC Official (Republic of China)
- Thai Buddha

For example, suppose NLS_CALENDAR is set to "Japanese Imperial", the date format is "E YY-MM-DD". ("E" is the date format element for the abbreviated era name.) If the date is May 15, 1997, then the SYSDATE is displayed as follows:

```
SELECT SYSDATE FROM DUAL;
SYSDATE
--------
H 09-05-15
```

**Note:**

The value of the initialization parameter NLS_CALENDER is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. If the initialization parameter is not specified, the initial session value becomes GREGORIAN. This initial value is overridden by a client-side value if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined.

**See Also:**

*Oracle Database Globalization Support Guide* for a listing of available calendar systems

## NLS_COMP

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_COMP = { BINARY \| LINGUISTIC \| ANSI } |
| **Default value** | BINARY |
| **Modifiable** | ALTER SESSION |
| **Basic** | No |

NLS_COMP specifies the collation behavior of the database session.

### Values

- BINARY

  Normally, comparisons in the WHERE clause and in PL/SQL blocks is binary unless you specify the NLSSORT function.

- LINGUISTIC

  Comparisons for all SQL operations in the WHERE clause and in PL/SQL blocks should use the linguistic sort specified in the NLS_SORT parameter. To improve the performance, you can also define a linguistic index on the column for which you want linguistic comparisons.

- ANSI

  A setting of ANSI is for backward compatibility; in general, you should set NLS_COMP to LINGUISTIC

---

**Note:**

Unless you explicitly set the value for NLS_COMP in your initialization parameter file, a default value of NULL is shown in the following views: V$PARAMETER, V$SYSTEM_PARAMETER, V$PARAMETER2, V$SYSTEM_PARAMETER2, and NLS_INSTANCE_PARAMETERS. However, the actual default value, and behavior, is BINARY. Note that you cannot change the default to NULL, because NULL is not among the valid values.

---

### Examples

See *Oracle Database Globalization Support Guide* for examples of using this parameter.

---

**Note:**

The value of this initialization parameter NLS_COMP is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

---

## NLS_CURRENCY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_CURRENCY = *currency_symbol* |
| **Default value** | Derived from NLS_TERRITORY |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid character string, with a maximum of 10 bytes (not including null) |

| Property | Description |
|---|---|
| **Basic** | No |

NLS_CURRENCY specifies the string to use as the local currency symbol for the L number format element. The default value of this parameter is determined by NLS_TERRITORY.

**Note:**

The value of this initialization parameter NLS_CURRENCY is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

- *Oracle Database Globalization Support Guide* for more information about this parameter
- *Oracle Database SQL Language Reference* for information on number format elements

# NLS_DATE_FORMAT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_DATE_FORMAT = "*format*" |
| **Default value** | Derived from NLS_TERRITORY |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid date format mask but not exceeding a fixed length |
| **Basic** | No |

NLS_DATE_FORMAT specifies the default date format to use with the TO_CHAR and TO_DATE functions. The default value of this parameter is determined by NLS_TERRITORY.

The value of this parameter can be any valid date format mask, and the value must be surrounded by double quotation marks. For example:

```
NLS_DATE_FORMAT = "MM/DD/YYYY"
```

**Note:**

The value of this initialization parameter NLS_DATE_FORMAT is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

# NLS_DATE_LANGUAGE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_DATE_LANGUAGE = *language* |
| **Default value** | Derived from NLS_LANGUAGE |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid NLS_LANGUAGE value |
| **Basic** | No |

NLS_DATE_LANGUAGE specifies the language to use for the spelling of day and month names and date abbreviations (a.m., p.m., AD, BC) returned by the TO_DATE and TO_CHAR functions.

**Note:**

The value of this initialization parameter NLS_DATE_LANGUAGE is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

- *Oracle Database Globalization Support Guide* for more information about this parameter
- *Oracle Database SQL Language Reference* for information on the TO_DATE function.
- *Oracle Database SQL Language Reference* for information on the TO_CHAR function.

## NLS_DUAL_CURRENCY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_DUAL_CURRENCY = *currency_symbol* |
| **Default value** | Derived from NLS_TERRITORY |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid format name up to 10 characters |
| **Basic** | No |

NLS_DUAL_CURRENCY specifies the dual currency symbol (such as "Euro") for the territory. The default is the dual currency symbol defined in the territory of your current language environment.

**Note:**

The value of this initialization parameter NLS_DUAL_CURRENCY is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

## NLS_ISO_CURRENCY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_ISO_CURRENCY = *territory* |

| Property | Description |
|---|---|
| **Default value** | Derived from NLS_TERRITORY |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid NLS_TERRITORY value |
| **Basic** | No |

NLS_ISO_CURRENCY specifies the string to use as the international currency symbol for the C number format element.

Local currency symbols can be ambiguous. For example, a dollar sign ($) can refer to U.S. dollars or Australian dollars. ISO Specification 4217 1987-07-15 defines unique "international" currency symbols for the currencies of specific territories or countries.

**Note:**

The value of this initialization parameter NLS_ISO_CURRENCY is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

- *Oracle Database Globalization Support Guide* for more information about this parameter
- *Oracle Database SQL Language Reference* for information on number format elements

# NLS_LANGUAGE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_LANGUAGE = *language* |
| **Default value** | Operating system-dependent, derived from the NLS_LANG environment variable |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid language name |
| **Basic** | Yes |

NLS_LANGUAGE specifies the default language of the database. This language is used for messages, day and month names, symbols for AD, BC, a.m., and p.m., and the default sorting mechanism. This parameter also determines the default values of the parameters NLS_DATE_LANGUAGE and NLS_SORT.

**Note:**

The value of this initialization parameter NLS_LANGUAGE is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

### Examples

See these examples of using the NLS_LANGUAGE parameter:

- For an example of setting NLS_LANGUAGE to Italian, see *Oracle Database Globalization Support Guide*.
- For an example of overriding default values for NLS_LANGUAGE and NLS_TERRITORY during a session, see *Oracle Database Globalization Support Guide*.

**See Also:**

- *Oracle Database Globalization Support Guide* for more information about this parameter.
- *Oracle Database Globalization Support Guide* for a complete list of languages that can be specified using this parameter
- *Oracle Database Globalization Support Guide* for information on overriding the default values for this parameter
- Your operating system-specific Oracle documentation and the release notes for your country

## NLS_LENGTH_SEMANTICS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_LENGTH_SEMANTICS = *string*<br>Example: NLS_LENGTH_SEMANTICS = 'CHAR' |
| **Default value** | BYTE |
| **Modifiable** | ALTER SESSION |
| **Range of values** | BYTE \| CHAR |
| **Basic** | No |

The session-level value of NLS_LENGTH_SEMANTICS specifies the default length semantics to use for VARCHAR2 and CHAR table columns, user-defined object attributes, and PL/SQL variables in database objects created in the session. This default may be overridden by the explicit length semantics qualifiers BYTE and CHAR in column, attribute, and variable definitions.

The instance-level value of NLS_LENGTH_SEMANTICS provides a default for the session-level value if NLS_LENGTH_SEMANTICS it is not set explicitly by the database client through the NLS_LENGTH_SEMANTICS client environment variable (does not apply to JDBC Thin clients), or the ALTER SESSION SET NLS_LENGTH_SEMANTICS statement.

NCHAR, NVARCHAR2, CLOB, and NCLOB columns are always character-based.

Sessions logged in as SYS do not use the NLS_LENGTH_SEMANTICS parameter. They use BYTE length semantics for all created objects unless overridden by the explicit BYTE and CHAR qualifiers in object definitions (SQL DDL statements).

**Note:**

Oracle strongly recommends that you do NOT set the NLS_LENGTH_SEMANTICS parameter to CHAR in the instance or server parameter file. This may cause many existing installation scripts to unexpectedly create columns with character length semantics, resulting in run-time errors, including buffer overflows.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

## NLS_NCHAR_CONV_EXCP

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_NCHAR_CONV_EXCP = { TRUE \| FALSE } |
| **Default value** | FALSE |
| **Modifiable** | ALTER SESSION |
| **Basic** | No |

NLS_NCHAR_CONV_EXCP determines whether an error is reported when there is data loss during an implicit or explicit character type conversion between NCHAR/NVARCHAR2 and CHAR/VARCHAR2. The default value results in no error being reported.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

# NLS_NUMERIC_CHARACTERS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_NUMERIC_CHARACTERS = "*decimal_character group_separator*" |
| **Default value** | Derived from NLS_TERRITORY |
| **Modifiable** | ALTER SESSION |
| **Basic** | No |

NLS_NUMERIC_CHARACTERS specifies the characters to use as the group separator and decimal character. It overrides those characters defined implicitly by NLS_TERRITORY. The group separator separates integer groups (that is, thousands, millions, billions, and so on). The decimal separates the integer portion of a number from the decimal portion.

You can specify any character as the decimal or group separator. The two characters specified must be single-byte and must be different from each other. The characters cannot be any numeric character or any of the following characters: plus (+), minus sign (-), less than sign (<), greater than sign (>). Either character can be a space.

For example, if you want to specify a comma as the decimal character and a space as the group separator, you would set this parameter as follows:

```
NLS_NUMERIC_CHARACTERS = ", "
```

**Note:**

The value of this initialization parameter NLS_NUMERIC_CHARACTERS is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

# NLS_SORT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_SORT = { BINARY \| *linguistic_definition* } |
| **Default value** | Derived from NLS_LANGUAGE |

| Property | Description |
|---|---|
| **Modifiable** | ALTER SESSION |
| **Range of values** | BINARY or any valid linguistic definition name |
| **Basic** | No |

NLS_SORT specifies the collating sequence for character value comparison in various SQL operators and clauses, for example, ORDER BY, GROUP BY, comparison conditions (=, <>, <=, >=), IN, BETWEEN, LIKE, MIN/MAX, GREATEST/LEAST, and INSTR.

- If the value is BINARY, then comparison is based directly on byte values in the binary encoding of the character values being compared. The ordering depends on the character set of the compared values, which is either the database character set (for VARCHAR2, CHAR, LONG, and CLOB) or the national character set (for NVARCHAR2, NCHAR, and NCLOB).
- If the value is a named linguistic sort, then comparison is defined by this sort. A linguistic sort uses various rules to achieve ordering expected by speakers of one or more natural languages. This is usually the same ordering that is used in dictionaries and telephone directories in those languages.

The exact operators and query clauses that obey the NLS_SORT parameter depend on the value of the NLS_COMP parameter. If an operator or clause does not obey the NLS_SORT value, as determined by NLS_COMP, the collation used is BINARY.

The BINARY comparison is faster and uses less resources than any linguistic comparison but for text in a natural language, it does not provide ordering expected by users.

The value of NLS_SORT affects execution plans of queries. Because a standard index cannot be used as a source of values sorted in a linguistic order, an explicit sort operation must usually be performed instead of an index range scan. A functional index on the NLSSORT function may be defined to provide values sorted in a linguistic order and reintroduce the index range scan to the execution plan.

---

**Note:**

The value of the initialization parameter NLS_SORT is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overriden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

---

---

**See Also:**

- *Oracle Database Globalization Support Guide* for more information about this parameter and a current listing of values you can specify

---

# NLS_TERRITORY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `NLS_TERRITORY = territory` |
| **Default value** | Operating system-dependent |
| **Modifiable** | `ALTER SESSION` |
| **Range of values** | Any valid territory name |
| **Basic** | Yes |

`NLS_TERRITORY` specifies the name of the territory whose conventions are to be followed for day and week numbering.

This parameter also establishes the default date format, the default decimal character and group separator, and the default ISO and local currency symbols.

For information on these settings, see "NLS_DATE_FORMAT", "NLS_NUMERIC_CHARACTERS", "NLS_CURRENCY", and "NLS_ISO_CURRENCY".

**Note:**

The value of this initialization parameter `NLS_TERRITORY` is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the `NLS_LANG` client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

### Examples

For an example of overriding the default value for the `NLS_TERRITORY` parameter, see *Oracle Database Globalization Support Guide*.

**See Also:**

- *Oracle Database Globalization Support Guide* for a complete list of territories
- Your operating system-specific Oracle documentation for the territory-dependent default values for these parameters

# NLS_TIMESTAMP_FORMAT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `NLS_TIMESTAMP_FORMAT = "format"` |

| Property | Description |
|---|---|
| **Default value** | Derived from NLS_TERRITORY |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid datetime format mask |
| **Basic** | No |

NLS_TIMESTAMP_FORMAT defines the default timestamp format to use with the TO_CHAR and TO_TIMESTAMP functions.

The value must be surrounded by quotation marks as follows:

```
NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH:MI:SS.FF'
```

You can specify the value of NLS_TIMESTAMP_FORMAT by setting it in the initialization parameter file. You can specify its value for a client as a client environment variable.

You can also alter the value of NLS_TIMESTAMP_FORMAT by changing its value in the initialization parameter and then restarting the instance. To alter the value during a session use the ALTER SESSION SET statement.

**Note:**

The value of this initialization parameter NLS_TIMESTAMP_FORMAT is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

# NLS_TIMESTAMP_TZ_FORMAT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | NLS_TIMESTAMP_TZ_FORMAT = "*format*" |
| **Default value** | Derived from NLS_TERRITORY |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any valid datetime format mask |
| **Basic** | No |

NLS_TIMESTAMP_TZ_FORMAT defines the default timestamp with time zone format to use with the TO_CHAR and TO_TIMESTAMP_TZ functions.

The value must be surrounded by quotation marks as follows:

```
NLS_TIMESTAMP_TZ_FORMAT = 'YYYY-MM-DD HH:MI:SS.FF TZH:TZM'
```

You can specify the value of NLS_TIMESTAMP_TZ_FORMAT by setting it in the initialization parameter file. You can specify its value for a client as a client environment variable.

You can also alter the value of NLS_TIMESTAMP_TZ_FORMAT by changing its value in the initialization parameter and then restarting the instance. To alter the value during a session use the ALTER SESSION SET statement.

**Note:**

The value of this initialization parameter NLS_TIMESTAMP_TZ_FORMAT is used to initialize the session value of this parameter, which is the actual value referenced by the SQL query processing. This initial value is overridden by a client-side value if the client uses the Oracle JDBC driver or if the client is OCI-based and the NLS_LANG client setting (environment variable) is defined. The initialization parameter value is, therefore, usually ignored.

**See Also:**

*Oracle Database Globalization Support Guide* for more information about this parameter

# NONCDB_COMPATIBLE

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | No |
| **Range of values** | true \| false |
| **Basic** | No |

NONCDB_COMPATIBLE enables you to get behavior similar to a non-CDB when issuing SQL commands inside a PDB in a CDB.

Set this parameter if you are using a single PDB in your CDB configuration and you have legacy code that causes ORA-65040 when you run it in the PDB.

### Values

- TRUE

  Indicates the behavior for SQL statements will be like a non-CDB although the statements are issued in a PDB in a CDB.

- FALSE

  Indicates the behavior will for SQL statements will be like a CDB. This is the default value of the parameter.

There are some statements (such as ALTER DB BACKUP CONTROLFILE) which can be issued in a non-CDB, but in a CDB they must be issued in the root (because they affect the whole CDB) and would result in an error if issued in a PDB.

Some ALTER DATABASE or ALTER SYSTEM statements are not permitted inside a PDB, and they will fail if NONCDB_COMPATIBLE=FALSE is set in init.ora. However, these statements will succeed if NONCDB_COMPATIBLE=TRUE is set.

**See Also:**

- *Oracle Database Concepts* for an introduction to PDBs and CDBs
- *Oracle Database Administrator's Guide* for information about the ALTER SYSTEM statements that can be run in a PDB.

# O7_DICTIONARY_ACCESSIBILITY

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | true |
| **Modifiable** | No |
| **Range of values** | true \| false |
| **Basic** | No |

O7_DICTIONARY_ACCESSIBILITY controls restrictions on SYSTEM privileges. If the parameter is set to true, access to objects in the SYS schema is allowed (Oracle7 behavior). The false setting ensures that system privileges that allow access to objects in "any schema" do not allow access to objects in the SYS schema.

For example, if O7_DICTIONARY_ACCESSIBILITY is set to false, then the SELECT ANY TABLE privilege allows access to views or tables in any schema except the SYS schema (data dictionary tables cannot be accessed). If O7_DICTIONARY_ACCESSIBILITY is set to false, then to access objects in the SYS schema, the user should have SELECT ANY DICTIONARY system privilege or the user should have been granted SELECT object privilege on the specific objects. The system privilege EXECUTE ANY PROCEDURE allows access on the procedures in any schema except the SYS schema.

If this parameter is set to false and you need to access objects in the SYS schema, then you must be granted explicit object privileges. The following roles, which can be granted to the database administrator, also allow access to dictionary objects:

- SELECT_CATALOG_ROLE
- EXECUTE_CATALOG_ROLE
- DELETE_CATALOG_ROLE

**Note:**

The DELETE_CATALOG_ROLE role is deprecated in Oracle Database 12*c* Release 1 (12.1.0.1).

**See Also:**

- *Oracle Database SQL Language Reference* for information on granting roles
- *Oracle Database 2 Day + Security Guide* for more information about enabling data dictionary protection using this parameter

# OBJECT_CACHE_MAX_SIZE_PERCENT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 10 |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM ... DEFERRED |
| **Range of values** | 0 to operating system-dependent maximum |
| **Basic** | No |

The **object cache** is a memory block on the client that allows applications to store entire objects and to navigate among them without round trips to the server. OBJECT_CACHE_MAX_SIZE_PERCENT specifies the percentage of the optimal cache size that the session object cache can grow past the optimal size. The maximum size is equal to the optimal size plus the product of this percentage and the optimal size. When the cache size exceeds this maximum size, the system will attempt to shrink the cache to the optimal size.

**See Also:**

- "OBJECT_CACHE_OPTIMAL_SIZE" for a description of the object cache
- *Pro*C/C++ Programmer's Guide* and *Oracle Call Interface Programmer's Guide* for information on precompiler use of the object cache

# OBJECT_CACHE_OPTIMAL_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 102400 (100K) |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM ... DEFERRED |
| **Range of values** | 10 KB to operating system-dependent maximum |

| Property | Description |
|---|---|
| **Basic** | No |

The **object cache** is a memory block on the client that allows applications to store entire objects and to navigate among them without round trips to the server. `OBJECT_CACHE_OPTIMAL_SIZE` specifies (in bytes) the size to which the session object cache is reduced when the size of the cache exceeds the maximum size.

**See Also:**

*Pro*C/C++ Programmer's Guide* and *Oracle Call Interface Programmer's Guide* for information on precompiler use of the object cache

# OLAP_PAGE_POOL_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `OLAP_PAGE_POOL_SIZE = integer [K \| M \| G]` |
| **Default value** | `0` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM ... DEFERRED` |
| **Range of values** | `0` to 2 GB |
| **Basic** | No |

`OLAP_PAGE_POOL_SIZE` specifies (in bytes) the size of the OLAP page pool.

**See Also:**

*Oracle OLAP User's Guide* for more information about the OLAP option for Oracle Database

# OPEN_CURSORS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `50` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 0 to 65535 |
| **Basic** | Yes |

`OPEN_CURSORS` specifies the maximum number of open cursors (handles to private SQL areas) a session can have at once. You can use this parameter to prevent a session from opening an excessive number of cursors.

It is important to set the value of OPEN_CURSORS high enough to prevent your application from running out of open cursors. The number will vary from one application to another. Assuming that a session does not open the number of cursors specified by OPEN_CURSORS, there is no added overhead to setting this value higher than actually needed.

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information on setting this parameter
- Your operating system-specific Oracle documentation for the range of values

# OPEN_LINKS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | No |
| **Range of values** | 0 to 255 |
| **Basic** | No |

OPEN_LINKS specifies the maximum number of concurrent open connections to remote databases in one session. These connections include database links, as well as external procedures and cartridges, each of which uses a separate process.

Oracle counts one open link for the following:

- For each user that references a public or private database link
- For each external procedure or cartridge connection when it is executed for the first time

Both types of connections close when the session ends. You can also close a database link connection explicitly by issuing an ALTER SESSION CLOSE DATABASE LINK statement.

You should set this parameter to allow for the external procedure and cartridge connections expected during the session plus the number of databases referred to in typical distributed transactions (that is, a single SQL statement that references multiple databases), so that all the databases can be open to execute the statement. For example, if queries alternately access databases A, B, and C, and OPEN_LINKS is set to 2, time will be lost waiting while one connection is broken and another made. Increase the value if many different databases are accessed over time.

This parameter refers only to connections used for distributed transactions. Direct connections to a remote database specified as an application connects are not counted.

If you set OPEN_LINKS to 0, then no distributed transactions are allowed.

**See Also:**

- "OPEN_LINKS_PER_INSTANCE" for information on setting open connections globally for a database instance
- *Oracle Streams Replication Administrator's Guide* for more information about using this parameter in an Oracle Streams environment

# OPEN_LINKS_PER_INSTANCE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 4 |
| **Modifiable** | No |
| **Range of values** | 0 to 4294967295 (4 GB -1) |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`OPEN_LINKS_PER_INSTANCE` specifies the maximum number of migratable open connections globally for each database instance. XA transactions use migratable open connections so that the connections are cached after a transaction is committed. Another transaction can use the connection, provided the user who created the connection is the same as the user who owns the transaction.

`OPEN_LINKS_PER_INSTANCE` is different from `OPEN_LINKS`, which indicates the number of connections from a session. The `OPEN_LINKS` parameter is not applicable to XA applications.

**See Also:**

- "OPEN_LINKS"
- *Oracle Database Development Guide* for more information about using this parameter in Oracle XA applications

# OPTIMIZER_ADAPTIVE_FEATURES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true | false` |
| **Basic** | No |

OPTIMIZER_ADAPTIVE_FEATURES enables or disables all of the adaptive optimizer features, including adaptive plan (adaptive join methods and bitmap plans), automatic re-optimization, SQL plan directives, and adaptive distribution methods.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about adaptive plans
- *Oracle Database SQL Tuning Guide* for more information about SQL plan directives

## OPTIMIZER_ADAPTIVE_REPORTING_ONLY

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | FALSE |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | TRUE \| FALSE |
| **Basic** | No |

OPTIMIZER_ADAPTIVE_REPORTING_ONLY controls reporting-only mode for adaptive optimizatons.

When OPTIMIZER_ADAPTIVE_REPORTING_ONLY is set to FALSE, reporting-only mode is off, and the adaptive optimizations are enabled as usual.

When OPTIMIZER_ADAPTIVE_REPORTING_ONLY is set to TRUE, adaptive optimizations run in reporting-only mode. With this setting, the information required for an adaptive optimization is gathered, but no action is taken to change the plan. For instance, an adaptive plan will always choose the default (optimizer-chosen) plan, but information is collected on what plan to adapt to in non-reporting mode. This information can be viewed in the adaptive plan report.

This parameter affects only adaptive optimizations that are enabled.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about adaptive plans and automatic reoptimization
- *Oracle Database SQL Tuning Guide* for information on controlling adaptive optimization

## OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |

| Property | Description |
|---|---|
| **Default value** | `true` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES` enables or disables the automatic recognition of repeatable SQL statements, as well as the generation of SQL plan baselines for such statements.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information about the optimizer

# OPTIMIZER_DYNAMIC_SAMPLING

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | If `OPTIMIZER_FEATURES_ENABLE` is set to `10.0.0` or higher, then `2`<br>If `OPTIMIZER_FEATURES_ENABLE` is set to `9.2.0`, then `1`<br>If `OPTIMIZER_FEATURES_ENABLE` is set to `9.0.1` or lower, then `0` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `0` to `11` |
| **Basic** | No |

`OPTIMIZER_DYNAMIC_SAMPLING` controls both when the database gathers dynamic statistics, and the size of the sample that the optimizer uses to gather the statistics.

**Note:**

Dynamic statistics were called dynamic sampling in releases before Oracle Database 12*c*.

When this parameter is set to `11`, the optimizer will use dynamic statistics to verify cardinality estimates for all SQL operators, and it will determine an internal time limit to spend verifying the estimates.

There are cases where the optimizer will automatically decide to use `11`, for example:

- The query will run in parallel.
- The query was executed before and its history is available (from the cursor cache, Automatic Workload Repository, or the SQL Management Base).

If the value of OPTIMIZER_DYNAMIC_SAMPLING is set to 11, the OPTIMIZER_FEATURES_ENABLE setting has no effect on the OPTIMIZER_DYNAMIC_SAMPLING setting.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information on setting this parameter

# OPTIMIZER_FEATURES_ENABLE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | OPTIMIZER_FEATURES_ENABLE = { 8.0.0 \| 8.0.3 \| 8.0.4 \| 8.0.5 \| 8.0.6 \| 8.0.7 \| 8.1.0 \| 8.1.3 \| 8.1.4 \| 8.1.5 \| 8.1.6 \| 8.1.7 \| 9.0.0 \| 9.0.1 \| 9.2.0 \| 9.2.0.8 \| 10.1.0 \| 10.1.0.3 \| 10.1.0.4 \| 10.1.0.5 \| 10.2.0.1 \| 10.2.0.2 \| 10.2.0.3 \| 10.2.0.4 \| 10.2.0.5 \| 11.1.0.6 \| 11.1.0.7 \| 11.2.0.1 \| 12.1.0.1 } |
| **Default value** | 12.1.0.1 |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

OPTIMIZER_FEATURES_ENABLE acts as an umbrella parameter for enabling a series of optimizer features based on an Oracle release number.

For example, if you upgrade your database from release 10.1 to release 11.1, but you want to keep the release 10.1 optimizer behavior, you can do so by setting this parameter to 10.1.0. At a later time, you can try the enhancements introduced in releases up to and including release 11.1 by setting the parameter to 11.1.0.6.

Table 1-6 describes some of the optimizer features that are enabled when you set the OPTIMIZER_FEATURES_ENABLE parameter to a 9.0 release or a 9.2 release.

Table 1-7 describes some of the optimizer features that are enabled when you set the OPTIMIZER_FEATURES_ENABLE parameter to a 10.1 release or a 10.2 release.

Table 1-8 describes some of the optimizer features that are enabled when you set the OPTIMIZER_FEATURES_ENABLE parameter to a 11.1 release.

Table 1-9 describes some of the optimizer features that are enabled when you set the OPTIMIZER_FEATURES_ENABLE parameter to a 12.1 release.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information about the optimizer and for information about the features listed in the following tables

***Table 1-6 Optimizer Features for Oracle9I Releases***

| Features | 9.0.0 | 9.0.1 | 9.2.0 |
|---|---|---|---|
| Index fast full scan | X | X | X |
| Consideration of bitmap access to paths for tables with only B-tree indexes | X | X | X |
| Complex view merging | X | X | X |
| Peeking into user-defined bind variables | X | X | X |
| Index joins | X | X | X |

***Table 1-7 Optimizer Features for Oracle Database 10g Releases***

| Features | 10.1.0 | 10.1.0.3 | 10.1.0.4 | 10.1.0.5 | 10.2.0.1 | 10.2.0.2 |
|---|---|---|---|---|---|---|
| Index fast full scan | X | X | X | X | X | X |
| Consideration of bitmap access to paths for tables with only B-tree indexes | X | X | X | X | X | X |
| Complex view merging | X | X | X | X | X | X |
| Peeking into user-defined bind variables | X | X | X | X | X | X |
| Index joins | X | X | X | X | X | X |
| Dynamic sampling | X | X | X | X | X | X |
| Query rewrite enables | X | X | X | X | X | X |
| Skip unusable indexes | X | X | X | X | X | X |
| Automatically compute index statistics as part of creation | X | X | X | X | X | X |
| Cost-based query transformations | X | X | X | X | X | X |
| Allow rewrites with multiple MVs and/or base tables | | | | | X | X |

***Table 1-8 Optimizer Features for Oracle Database 11g Releases***

| Features | 11.1.0.6 |
|---|---|
| Enhanced Bind Peeking | X |
| Use extended statistics to estimate selectivity | X |
| Use native implementation for full outer joins | X |
| Partition pruning using join filtering | X |
| Group by placement optimization | X |
| Null aware antijoins | X |

***Table 1-9 Optimizer Features for Oracle Database 12c Releases***

| Features | 12.1.0.1 |
|---|---|
| Adaptive Execution Plans | X |

| Features | 12.1.0.1 |
|---|---|
| Online statistics gathering for bulk loads | X |
| Session level statistics for Global Temporary Tables | X |
| Multi-table left outer joins | X |
| Lateral views | X |
| Batch table access by rowid | X |
| Null accepting semi joins | X |
| Scalar subquery unnesting | X |
| Conversion of joins that produce unnecessary duplicates to semi-joins | X |
| Parallel Union and Parallel Union All operations | X |
| Enhance Auto DOP | X |

# OPTIMIZER_INDEX_CACHING

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | 0 to 100 |
| **Basic** | No |

`OPTIMIZER_INDEX_CACHING` lets you adjust the behavior of cost-based optimization to favor nested loops joins and IN-list iterators.

The cost of executing an index using an IN-list iterator or of executing a nested loops join when an index is used to access the inner table depends on the caching of that index in the buffer cache. The amount of caching depends on factors that the optimizer cannot predict, such as the load on the system and the block access patterns of different users.

You can modify the optimizer's assumptions about index caching for nested loops joins and IN-list iterators by setting this parameter to a value between 0 and 100 to indicate the percentage of the index blocks the optimizer should assume are in the cache. Setting this parameter to a higher value makes nested loops joins and IN-list iterators look less expensive to the optimizer. As a result, it will be more likely to pick nested loops joins over hash or sort-merge joins and to pick indexes using IN-list iterators over other indexes or full table scans. The default for this parameter is 0, which results in default optimizer behavior.

**See Also:**

*Oracle Database SQL Language Reference* for additional information about this initialization parameter

# OPTIMIZER_INDEX_COST_ADJ

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `100` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | 1 to 10000 |
| **Basic** | No |

`OPTIMIZER_INDEX_COST_ADJ` lets you tune optimizer behavior for access path selection to be more or less index friendly—that is, to make the optimizer more or less prone to selecting an index access path over a full table scan.

The default for this parameter is 100 percent, at which the optimizer evaluates index access paths at the regular cost. Any other value makes the optimizer evaluate the access path at that percentage of the regular cost. For example, a setting of 50 makes the index access path look half as expensive as normal.

**Note:**

The adjustment does not apply to user-defined cost functions for domain indexes.

**See Also:**

*Oracle Database SQL Language Reference* for additional information about this initialization parameter

# OPTIMIZER_INMEMORY_AWARE

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | All instances should use the same value |

`OPTIMIZER_INMEMORY_AWARE` enables or disables all of the optimizer cost model enhancements for in-memory. Setting the parameter to `false` causes the optimizer to ignore the in-memory property of tables during the optimization of SQL statements. This behavior can also be achieved by setting the `OPTIMIZER_FEATURES_ENABLE` initialization parameter to values lower than 12.1.0.2.

**Note:**

This initialization parameter is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "OPTIMIZER_FEATURES_ENABLE"
- *Oracle Database Concepts* and *Oracle Database SQL Tuning Guide* for more information about the optimizer

# OPTIMIZER_MODE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `OPTIMIZER_MODE = { FIRST_ROWS_[1 \| 10 \| 100 \| 1000] \| FIRST_ROWS \| ALL_ROWS }` |
| **Default value** | `ALL_ROWS` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |

`OPTIMIZER_MODE` establishes the default behavior for choosing an optimization approach for the instance.

### Values

- `FIRST_ROWS_`*n*

  The optimizer uses a cost-based approach and optimizes with a goal of best response time to return the first *n* rows (where *n* = 1, 10, 100, 1000).

- `FIRST_ROWS`

  The optimizer uses a mix of costs and heuristics to find a best plan for fast delivery of the first few rows.

  `FIRST_ROWS` is available for backward compatibility and plan stability; use `FIRST_ROWS_`*n* instead.

- `ALL_ROWS`

  The optimizer uses a cost-based approach for all SQL statements in the session and optimizes with a goal of best throughput (minimum resource use to complete the entire statement).

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information on setting this parameter
- *Oracle Database Concepts* and *Oracle Database SQL Tuning Guide* for more information about the optimizer

# OPTIMIZER_SECURE_VIEW_MERGING

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values |

`OPTIMIZER_SECURE_VIEW_MERGING` enables the optimizer to use view merging to improve query performance without performing the checks that would otherwise be performed to ensure that view merging does not violate any security intentions of the view creator.

### Values

- `false`

  Oracle Database does not perform security checks that may prevent view merging and predicate move-around.

- `true`

  Oracle Database performs checks to ensure that view merging and predicate move-around do not violate any security intentions of the view creator.

To enable the optimizer to use view merging for any query issued by a user, you must grant the `MERGE ANY VIEW` privilege to the user. Grant the `MERGE VIEW` privilege to a user on specific views to enable the optimizer to use view merging for queries on these views. These privileges are required only under specific conditions, such as when a view is not merged because the security checks fail.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information about view merging

# OPTIMIZER_USE_INVISIBLE_INDEXES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true | false` |
| **Basic** | No |

`OPTIMIZER_USE_INVISIBLE_INDEXES` enables or disables the use of invisible indexes.

### Values

- `true`

  Invisible indexes are treated as visible (normal) indexes.

- `false`

  Invisible indexes will not be considered by the optimizer but will still be maintained by DML operations.

# OPTIMIZER_USE_PENDING_STATISTICS

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true | false` |
| **Basic** | No |

`OPTIMIZER_USE_PENDING_STATISTICS` specifies whether the optimizer uses pending statistics when compiling SQL statements.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information on setting this parameter

# OPTIMIZER_USE_SQL_PLAN_BASELINES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |

| Property | Description |
|---|---|
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`OPTIMIZER_USE_SQL_PLAN_BASELINES` enables or disables the use of SQL plan baselines stored in SQL Management Base. When enabled, the optimizer looks for a SQL plan baseline for the SQL statement being compiled. If one is found in SQL Management Base, then the optimizer will cost each of the baseline plans and pick one with the lowest cost.

**See Also:**

- *Oracle Database SQL Tuning Guide* for information on enabling automatic initial plan capture
- *Oracle Database SQL Tuning Guide* for information about configuring the capture and use of SQL plan baselines

# OS_AUTHENT_PREFIX

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `OS_AUTHENT_PREFIX = authentication_prefix` |
| **Default value** | `OPS$` |
| **Modifiable** | No |
| **Basic** | No |

`OS_AUTHENT_PREFIX` specifies a prefix that Oracle Database uses to authenticate users attempting to connect to the server. Oracle Database concatenates the value of this parameter to the beginning of the user's operating system account name. When a connection request is attempted, Oracle Database compares the prefixed username with Oracle user names in the database.

The default value of this parameter is `OPS$` for backward compatibility with previous versions. However, you might prefer to set the prefix value to "" (a null string), thereby eliminating the addition of any prefix to operating system account names.

**Note:**

The text of the `OS_AUTHENT_PREFIX` parameter is case sensitive on some operating systems.

**See Also:**

- *Oracle Database Security Guide* for more information on setting this parameter
- Your operating system-specific Oracle documentation for the default value

# OS_ROLES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`OS_ROLES` determines whether Oracle or the operating system identifies and manages the roles of each username.

### Values

- `TRUE`

  The operating system completely manages the role grants for all database usernames. When a user attempts to create a session, the username's security domain is initialized using the roles identified by the operating system. A user can subsequently enable as many roles identified by the operating system as specified by the parameter `MAX_ENABLED_ROLES`.

  Revocation by Oracle of roles granted by the operating system is ignored, as are any roles previously granted by Oracle.

- `FALSE`

  Oracle identifies and manages the roles.

**See Also:**

- *Oracle Database Administrator's Guide* and *Oracle Database Enterprise User Security Administrator's Guide* for more more information on roles and on setting this parameter
- "REMOTE_OS_ROLES"

# PARALLEL_ADAPTIVE_MULTI_USER

| Property | Description |
|---|---|
| **Parameter type** | Boolean |

| Property | Description |
|---|---|
| **Default value** | `true` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`PARALLEL_ADAPTIVE_MULTI_USER`, when set to `true`, enables an adaptive algorithm designed to improve performance in multiuser environments that use parallel execution. The algorithm automatically reduces the requested degree of parallelism based on the system load at query startup time. The effective degree of parallelism is based on the default degree of parallelism, or the degree from the table or hints, divided by a reduction factor.

The algorithm assumes that the system has been tuned for optimal performance in a single-user environment.

Tables and hints use the default degree of parallelism.

**See Also:**

*Oracle Database SQL Language Reference* for more information about optimizer hints

# PARALLEL_AUTOMATIC_TUNING

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

**Note:**

The `PARALLEL_AUTOMATIC_TUNING` parameter is deprecated. It is retained for backward compatibility only.

When `PARALLEL_AUTOMATIC_TUNING` is set to `true`, Oracle determines the default values for parameters that control parallel execution. In addition to setting this parameter, you must specify the `PARALLEL` clause for the target tables in the system. Oracle then tunes all subsequent parallel operations automatically.

If you used parallel execution in a previous release and are now enabling `PARALLEL_AUTOMATIC_TUNING`, then you should reduce the amount of memory allocated from the shared pool to account for the decreased demand on that pool. This

memory will now be allocated from the large pool, and will be computed automatically if LARGE_POOL_SIZE is not specified.

As part of the automatic tuning, Oracle will enable the PARALLEL_ADAPTIVE_MULTI_USER parameter. You can override any of the system-provided defaults if desired.

# PARALLEL_DEGREE_LIMIT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PARALLEL_DEGREE_LIMIT = { CPU \| IO \| *integer* } |
| **Default value** | CPU |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

With automatic degree of parallelism, Oracle automatically decides whether a statement should execute in parallel and what degree of parallelism the statement should use. The optimizer automatically determines the degree of parallelism for a statement based on the resource requirements of the statement. However, the optimizer will limit the degree of parallelism used to ensure parallel server processes do not flood the system. This limit is enforced by PARALLEL_DEGREE_LIMIT.

### Values

- CPU

  The maximum degree of parallelism is limited by the number of CPUs in the system. The formula used to calculate the limit is PARALLEL_THREADS_PER_CPU * CPU_COUNT * the number of instances available (by default, all the opened instances on the cluster but can be constrained using PARALLEL_INSTANCE_GROUP or service specification). This is the default.

- IO

  The maximum degree of parallelism the optimizer can use is limited by the I/O capacity of the system. The value is calculated by dividing the total system throughput by the maximum I/O bandwidth per process. You must run the DBMS_RESOURCE_MANAGER.CALIBRATE_IO procedure on the system to use the IO setting. This procedure will calculate the total system throughput and the maximum I/O bandwidth per process.

- *integer*

  A numeric value for this parameter specifies the maximum degree of parallelism the optimizer can choose for a SQL statement when automatic degree of parallelism is active. Automatic degree of parallelism is only enabled if PARALLEL_DEGREE_POLICY is set to ADAPTIVE, AUTO, or LIMITED.

**See Also:**

- *Oracle Database VLDB and Partitioning Guide* for information about automatic degree of parallelism
- *Oracle Database PL/SQL Packages and Types Reference* for information on the `DBMS_RESOURCE_MANAGER.CALIBRATE_IO` procedure

# PARALLEL_DEGREE_POLICY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `PARALLEL_DEGREE_POLICY = { MANUAL \| LIMITED \| AUTO \| ADAPTIVE }` |
| **Default value** | `MANUAL` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |

`PARALLEL_DEGREE_POLICY` specifies whether automatic degree of parallelism, statement queuing, and in-memory parallel execution will be enabled.

### Values

**Note:**

Automatic degree of parallelism will be enabled regardless of the value of `PARALLEL_DEGREE_POLICY` if a `PARALLEL` hint is used at the SQL statement level.

- `MANUAL`

  Disables automatic degree of parallelism, statement queuing, and in-memory parallel execution. This reverts the behavior of parallel execution to what it was prior to Oracle Database 11*g* Release 2 (11.2). This is the default.

- `LIMITED`

  Enables automatic degree of parallelism for some statements but statement queuing and in-memory Parallel Execution are disabled. Automatic degree of parallelism is only applied to those statements that access tables or indexes decorated explicitly with the `DEFAULT` degree of parallelism using the `PARALLEL` clause. Statements that do not access any tables or indexes decorated with the `DEFAULT` degree of parallelism will retain the `MANUAL` behavior.

- `AUTO`

  Enables automatic degree of parallelism, statement queuing, and in-memory parallel execution.

- `ADAPTIVE`

This value enables automatic degree of parallelism, statement queuing and in-memory parallel execution, similar to the AUTO value. In addition, performance feedback is enabled. Performance feedback helps to improve the degree of parallelism automatically chosen for repeated SQL statements. After the initial execution of a statement, the degree of parallelism chosen by the optimizer is compared to the degree of parallelism computed based on the actual execution performance. If they vary significantly, then the statement is marked for re-parse and the initial execution performance statistics (for example, CPU-time) are provided as feedback for subsequent executions. The optimizer uses the initial execution performance statistics to better determine a degree of parallelism for subsequent executions.

**See Also:**

- *Oracle Database SQL Language Reference* for information about PARALLEL hints
- *Oracle Database VLDB and Partitioning Guide* for information about automatic degree of parallelism

# PARALLEL_EXECUTION_MESSAGE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | Operating system-dependent |
| **Modifiable** | No |
| **Range of values** | Minimum: 2148<br>Maximum: 65536, but some operating systems may have a smaller value |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must have the same value. |

PARALLEL_EXECUTION_MESSAGE_SIZE specifies the size of messages used for parallel execution (formerly referred to as parallel query, PDML, Parallel Recovery, replication).

On most platforms, the default value is as follows:

- 16384 bytes if COMPATIBLE is set to 11.2.0 or higher
- 4096 bytes if COMPATIBLE is less than 11.2.0 and PARALLEL_AUTOMATIC_TUNING is set to true
- 2148 bytes if COMPATIBLE is less than 11.2.0 and PARALLEL_AUTOMATIC_TUNING is set to false

The default value is adequate for most applications. Larger values require a larger shared pool. Larger values result in better performance at the cost of higher memory use. For this reason, replication gets no benefit from increasing the size.

**Note:**

When PARALLEL_AUTOMATIC_TUNING is set to TRUE, message buffers are allocated out of the large pool. In this case, the default is generally higher. Note that the PARALLEL_AUTOMATIC_TUNING parameter has been deprecated.

**See Also:**

*Oracle Database VLDB and Partitioning Guide* to learn how this parameter affects memory consumption for parallel operations, including parallel execution

## PARALLEL_FORCE_LOCAL

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | true \| false |
| **Basic** | No |

PARALLEL_FORCE_LOCAL controls parallel execution in an Oracle RAC environment. By default, the parallel server processes selected to execute a SQL statement can operate on any or all Oracle RAC nodes in the cluster. By setting PARALLEL_FORCE_LOCAL to true, the parallel server processes are restricted so that they can only operate on the same Oracle RAC node where the query coordinator resides (the node on which the SQL statement was executed).

**See Also:**

*Oracle Database VLDB and Partitioning Guide* for more information about this parameter

## PARALLEL_INSTANCE_GROUP

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PARALLEL_INSTANCE_GROUP = *service_name* \| *group_name* |
| **Default value** | There is no default value; parallel execution is enabled across all currently active instances. |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | Any service name or any group name specified in the INSTANCE_GROUPS parameter of any active instance |

| Property | Description |
|---|---|
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

PARALLEL_INSTANCE_GROUP is an Oracle RAC parameter that you can specify in parallel mode only. Used in conjunction with services or with the INSTANCE_GROUPS parameter, it lets you restrict parallel query operations to a limited number of instances. Note that the INSTANCE_GROUPS parameter has been deprecated.

This parameter identifies the parallel instance group Oracle will use for spawning parallel execution processes. If used in conjunction with services, then parallel operations will spawn parallel execution processes only on instances defined in the service. If used in conjunction with INSTANCE_GROUPS, then parallel operations will spawn parallel execution processes only on instances that specify a matching group in their INSTANCE_GROUPS parameter.

If the value assigned to PARALLEL_INSTANCE_GROUP is the name of a service or group that does not exist, then the operation runs serially. No parallelism is used.

# PARALLEL_IO_CAP_ENABLED

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | true |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | true \| false |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

**Note:**

The PARALLEL_IO_CAP_ENABLED parameter is deprecated. It is retained for backward compatibility only. It is replaced by the PARALLEL_DEGREE_LIMIT parameter when set to IO.

PARALLEL_IO_CAP_ENABLED specifies whether Oracle caps the default degree of parallelism to no greater than what the I/O system can support. This new value is calculated based on the results of the resource manager's I/O calibration package.

If PARALLEL_IO_CAP_ENABLED is set to true and the database I/O capacity has been calibrated, then Oracle will reduce the default degree of parallelism if the I/O capacity cannot support the available number of CPUs. To calibrate the I/O capacity, use the DBMS_RESOURCE_MANAGER.CALIBRATE_IO procedure to measure the system's I/O capacity. Oracle then computes the degree of parallelism so as not to exceed the system's I/O capacity.

**See Also:**

"PARALLEL_DEGREE_LIMIT"

# PARALLEL_MAX_SERVERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | PARALLEL_THREADS_PER_CPU * CPU_COUNT * *concurrent_parallel_users* * 5 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 to 32767 |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

**Note:**

This parameter applies to parallel execution in exclusive mode as well as in a Real Application Clusters environment.

PARALLEL_MAX_SERVERS specifies the maximum number of parallel execution processes and parallel recovery processes for an instance. As demand increases, Oracle Database increases the number of processes from the number created at instance startup up to this value.

The number of concurrent parallel users running at default degree of parallelism on an instance depends on the memory initialization parameter settings for the instance. For example, if the MEMORY_TARGET or SGA_TARGET initialization parameter is set, then the number of concurrent_parallel_users = 4. If neither MEMORY_TARGET or SGA_TARGET is set, then PGA_AGGREGATE_TARGET is examined. If a value is set for PGA_AGGREGATE_TARGET, then concurrent_parallel_users = 2. If a value is not set for PGA_AGGREGATE_TARGET, then concurrent_parallel_users = 1.

If you set this parameter too low, then some queries may not have a parallel execution process available to them during query processing. If you set it too high, then memory resource shortages may occur during peak periods, which can degrade performance.

**Note:**

The database system always reserves a certain number of reserved processes.

The default value of the PARALLEL_MAX_SERVERS initialization parameter is set to the lower of these two values:

- The default value of PARALLEL_MAX_SERVERS determined using the calculation in the table above
- The value of the PROCESSES initialization parameter minus the number of reserved processes

When the PARALLEL_MAX_SERVERS parameter is set to a value lower than the value in the table above, the lower default value enables the database to start service processes and allows user processes to connect to the database.

**See Also:**

*Oracle Database SQL Tuning Guide* for more information about parallel execution

# PARALLEL_MIN_PERCENT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | ALTER SESSION |
| **Range of values** | 0 to 100 |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

PARALLEL_MIN_PERCENT lets you specify the minimum percentage of the requested number of parallel execution processes required for parallel execution. This parameter controls the behavior for parallel operations when parallel statement queuing is not enabled (when PARALLEL_DEGREE_POLICY is set to manual or limited). It ensures that an operation always gets a minimum percentage of parallel execution servers or errors out. Setting this parameter ensures that parallel operations will not execute unless adequate resources are available. The default value of 0 means that no minimum percentage of processes has been set.

Consider the following settings:

```
PARALLEL_MIN_PERCENT = 50
PARALLEL_MIN_SERVERS = 5
PARALLEL_MAX_SERVERS = 10
```

If 8 of the 10 parallel execution processes are busy, only 2 processes are available. If you then request a query with a degree of parallelism of 8, the minimum 50% will not be met.

You can use this parameter with PARALLEL_ADAPTIVE_MULTI_USER. In a multi-user environment, an individual user or application can set PARALLEL_MIN_PERCENT to a minimum value until sufficient resources are available on the system and an acceptable degree of parallelism is returned.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about parallel execution
- "PARALLEL_DEGREE_POLICY", "PARALLEL_MAX_SERVERS", "PARALLEL_MIN_SERVERS", and "PARALLEL_ADAPTIVE_MULTI_USER"

# PARALLEL_MIN_SERVERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | CPU_COUNT * PARALLEL_THREADS_PER_CPU * 2 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | Default value to the value of PARALLEL_MAX_SERVERS |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

**Note:**

This parameter applies to parallel execution in exclusive mode as well as in an Oracle Real Application Clusters environment.

PARALLEL_MIN_SERVERS is the number of parallel execution processes Oracle creates when the instance is started. These processes will be kept alive to service parallel statements.

**Note:**

When the PROCESSES initialization parameter is set to a value that is lower than the documented default value for the PARALLEL_MIN_SERVERS parameter in the table above, the database sets the default value of PARALLEL_MIN_SERVERS to a value that is lower than the documented default value. The lower default value enables the database to start service processes and allows user processes to connect to the database.

**See Also:**

*Oracle Database Administrator's Guide* for more information about parallel execution servers

## PARALLEL_MIN_TIME_THRESHOLD

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PARALLEL_MIN_TIME_THRESHOLD = { AUTO \| *integer* } |
| **Default value** | AUTO |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

PARALLEL_MIN_TIME_THRESHOLD specifies the minimum execution time a statement should have before the statement is considered for automatic degree of parallelism. By default, this is set to 10 seconds. Automatic degree of parallelism is only enabled if PARALLEL_DEGREE_POLICY is set to ADAPTIVE, AUTO, or LIMITED.

If all tables referenced by a SQL statement use In-Memory Column Store (IM column store), then PARALLEL_MIN_TIME_THRESHOLD defaults to 1.

**See Also:**

*Oracle Database VLDB and Partitioning Guide* for information about automatic degree of parallelism

## PARALLEL_SERVERS_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | PARALLEL_THREADS_PER_CPU * CPU_COUNT * *concurrent_parallel_users* * 2 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 to PARALLEL_MAX_SERVERS |
| **Basic** | No |

PARALLEL_SERVERS_TARGET specifies the number of parallel server processes allowed to run parallel statements before statement queuing will be used. When the parameter PARALLEL_DEGREE_POLICY is set to ADAPTIVE or AUTO, Oracle will queue SQL statements that require parallel execution, if the necessary parallel server processes are not available. Statement queuing will begin once the number of parallel server processes active on the system is equal to or greater than PARALLEL_SERVERS_TARGET.

**Note:**

Consumer groups that have been marked with the PARALLEL_STMT_CRITICAL directive are allowed to bypass the parallel statement queue, and therefore may drive the total number of active parallel server processes beyond PARALLEL_SERVERS_TARGET.

By default, PARALLEL_SERVERS_TARGET is set lower than the maximum number of parallel server processes allowed on the system (PARALLEL_MAX_SERVERS) to ensure each parallel statement will get all of the parallel server resources required and to prevent overloading the system with parallel server processes.

The number of concurrent parallel users running at default degree of parallelism on an instance depends on the memory initialization parameter settings for the instance. For example, if the MEMORY_TARGET or SGA_TARGET initialization parameter is set, then the number of concurrent_parallel_users = 4. If neither MEMORY_TARGET or SGA_TARGET is set, then PGA_AGGREGATE_TARGET is examined. If a value is set for PGA_AGGREGATE_TARGET, then concurrent_parallel_users = 2. If a value is not set for PGA_AGGREGATE_TARGET, then concurrent_parallel_users = 1.

Note that all serial (non-parallel) statements will execute immediately even if statement queuing has been activated.

**See Also:**

- "PARALLEL_DEGREE_POLICY"
- "PARALLEL_MAX_SERVERS"
- "MEMORY_TARGET"
- "SGA_TARGET"
- "PGA_AGGREGATE_TARGET"
- *Oracle Database VLDB and Partitioning Guide* for more information about this parameter

# PARALLEL_THREADS_PER_CPU

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | Operating system-dependent, usually 2 |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | Any nonzero number |
| **Basic** | No |

**Note:**

This parameter applies to parallel execution in exclusive mode as well as in a Real Application Clusters environment.

PARALLEL_THREADS_PER_CPU specifies the default degree of parallelism for the instance and determines the parallel adaptive and load balancing algorithms. The parameter describes the number of parallel execution processes or **threads** that a CPU can handle during parallel execution.

The default is platform-dependent and is usually adequate. You should decrease the value of this parameter if the system appears to be overloaded when a representative parallel query is executed. You should increase the value if the system is I/O bound.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about parallel execution
- *Oracle Database VLDB and Partitioning Guide* for information about how the optimizer automatically determines the degree of parallelism for a statement

# PDB_FILE_NAME_CONVERT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PDB_FILE_NAME_CONVERT = '*string1*' , '*string2*' , '*string3*' , '*string4*' , ...<br>Where:<br>• *string1* is the pattern of the existing filename<br>• *string2* is the pattern of the new filename<br>• *string3* is the pattern of the existing filename<br>• *string4* is the pattern of the new filename<br>You can use as many pairs of existing and new replacement strings as required. You can use single or double quotation marks.<br>The following are example settings that are acceptable:<br>PDB_FILE_NAME_CONVERT = '/dbs/t1/','/dbs/t1/s_','dbs/t2/ ','dbs/t2/s_' |
| **Default value** | There is no default value |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |
| **Oracle RAC** | You must set this parameter for every instance, and multiple instances must have the same value. |

PDB_FILE_NAME_CONVERT maps names of existing files to new file names when processing a CREATE PLUGGABLE DATABASE statement, as well as when processing the ENABLE PLUGGABLE DATABASE clause of the CREATE DATABASE statement, if

the `file_name_convert_clause` is not specified and Oracle Managed Files is not enabled.

File name patterns specified in this initialization parameter cannot match files or directories managed by Oracle Managed Files.

**See Also:**

*Oracle Database Administrator's Guide* for more information about this parameter

# PERMIT_92_WRAP_FORMAT

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances should use the same value |

`PERMIT_92_WRAP_FORMAT` allows Oracle Database release 9.2 wrapped versions of PL/SQL source text to be used in Oracle Database releases 10.2, 11.2, and 12.1 when this parameter is set to `true`.

Oracle recommends that wrapped files be created using the PL/SQL Wrapper utility from Oracle Database release 10 or later.

**See Also:**

*Oracle Database PL/SQL Language Reference* for more information about PL/SQL source text wrapping

# PGA_AGGREGATE_LIMIT

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `PGA_AGGREGATE_LIMIT = integer [K \| M \| G]` |
| **Default value** | By default, `PGA_AGGREGATE_LIMIT` is set to the greater of 2 GB, 200% of `PGA_AGGREGATE_TARGET`, and 3 MB times the `PROCESSES` parameter. It will be set below 200% of `PGA_AGGREGATE_TARGET` if it is larger than 90% of the physical memory size minus the total SGA size, but not below 100% of `PGA_AGGREGATE_TARGET`. |
| **Modifiable** | `ALTER SYSTEM` |

| Property | Description |
|---|---|
| **Range of values** | Oracle recommends that you do not set `PGA_AGGREGATE_LIMIT` below its default value. `PGA_AGGREGATE_LIMIT` cannot be set below its default value except in a text initialization parameter file (pfile) or binary server parameter file (spfile). If a value of 0 is specified, it means there is no limit to the aggregate PGA memory consumed by the instance. |
| **Basic** | No |

`PGA_AGGREGATE_LIMIT` specifies a limit on the aggregate PGA memory consumed by the instance.

There is no difference in behavior between `PGA_AGGREGATE_LIMIT` being explicitly set or being set to the default.

### Actions Taken When PGA_AGGREGATE_LIMIT is Exceeded

Parallel queries will be treated as a unit. First, the sessions that are using the most untunable memory will have their calls aborted. Then, if the total PGA memory usage is still over the limit, the sessions that are using the most untunable memory will be terminated.

SYS processes and background processes other than job queue processes will not be subjected to any of the actions described in this section. Instead, if they are using the most untunable memory, they will periodically write a brief summary of their PGA usage to a trace file.

**See Also:**

- "MEMORY_TARGET"
- "PGA_AGGREGATE_TARGET"
- *Oracle Database Performance Tuning Guide* for more information about this parameter

## PGA_AGGREGATE_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `PGA_AGGREGATE_TARGET = integer [K \| M \| G]` |
| **Default value** | `10` MB or 20% of the size of the SGA, whichever is greater |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Minimum: 10 MB<br>Maximum: 4096 GB - 1 |
| **Basic** | Yes |

PGA_AGGREGATE_TARGET specifies the target aggregate PGA memory available to all server processes attached to the instance. To set a hard limit for aggregate PGA memory, use the PGA_AGGREGATE_LIMIT parameter.

Setting PGA_AGGREGATE_TARGET to a nonzero value has the effect of automatically setting the WORKAREA_SIZE_POLICY parameter to AUTO. With this setting, SQL working areas used by memory-intensive SQL operators (such as sort, group-by, hash-join, bitmap merge, and bitmap create) will be automatically sized. A nonzero value for this parameter is the default since, unless you specify otherwise, Oracle sets it to 20% of the SGA or 10 MB, whichever is greater.

Setting PGA_AGGREGATE_TARGET to 0 automatically sets the WORKAREA_SIZE_POLICY parameter to MANUAL. With this setting, SQL working areas are sized using the *_AREA_SIZE parameters.

Oracle attempts to keep the amount of private memory below the target specified by this parameter by adapting the size of the working areas to private memory. When increasing the value of this parameter, you indirectly increase the memory allotted to working areas. Consequently, more memory-intensive operations are able to run fully in memory and fewer will work their way over to disk.

If Automatic Memory Management is enabled (MEMORY_TARGET is set to a positive value) and PGA_AGGREGATE_TARGET is also set to a positive value, the PGA_AGGREGATE_TARGET value acts as the minimum value for the size of the instance PGA.

**See Also:**

- "WORKAREA_SIZE_POLICY"
- *Oracle Database Performance Tuning Guide* for more information about configuring memory
- "PGA_AGGREGATE_LIMIT"

# PLSCOPE_SETTINGS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PLSCOPE_SETTINGS = IDENTIFIERS:{ NONE \| ALL } |
| **Default value** | IDENTIFIERS:NONE |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

PLSCOPE_SETTINGS controls the compile time collection, cross-reference, and storage of PL/SQL source code identifier data.

### Values

- IDENTIFIERS:NONE

  Disables collection of identifier data. This is the default.

- `IDENTIFIERS:ALL`

  Enables the collection of all source code identifier data.

`PLSCOPE_SETTINGS` can be set on a session, system, or per-library unit (`ALTER COMPILE`) basis. The current setting of `PLSCOPE_SETTINGS` for any library unit can be attained by querying the `*_PLSQL_OBJECT_SETTINGS` views. Any identifier data collected by setting this parameter can be accessed using the `*_IDENTIFIERS` views.

**See Also:**

- "ALL_PLSQL_OBJECT_SETTINGS," "DBA_PLSQL_OBJECT_SETTINGS," and "USER_PLSQL_OBJECT_SETTINGS"
- "ALL_IDENTIFIERS,""DBA_IDENTIFIERS," and "USER_IDENTIFIERS"
- *Oracle Database PL/SQL Language Reference* for more information about this parameter

# PLSQL_CCFLAGS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PLSQL_CCFLAGS = '<*v1*>:<*c1*>,<*v2*>:<*c2*>,...,<*vn*>:<*cn*>' |
| **Default value** | Empty string |
| **Modifiable** | `ALTER SESSION, ALTER SYSTEM` |
| **Range of values** | Any string literal that satisfies the internal syntax |
| **Basic** | No |
| **Examples** | `ALTER SESSION SET PLSQL_CCFLAGS = 'DeBug:TruE';`<br>`ALTER SESSION SET PLSQL_CCFLAGS = 'debug:TRUE';` |

`PLSQL_CCFLAGS` provides a mechanism that allows PL/SQL programmers to control conditional compilation of each PL/SQL library unit independently.

### Values

- <*vi*> has the form of an unquoted PL/SQL identifier. It is unrestricted and can be a reserved word or a keyword. The text is insensitive to case. Each one is known as a flag or flag name. Each <*vi*> can occur more than once in the string, each occurrence can have a different flag value, and the flag values can be of different kinds.
- <*ci*> is one of the following: a PL/SQL boolean literal, a `PLS_INTEGER` literal, or the literal NULL. The text is insensitive to case. Each one is known as a flag value and corresponds to a flag name.

You can define any allowable value for `PLSQL_CCFLAGS`. However, Oracle recommends that this parameter be used for controlling the conditional compilation of debugging or tracing code. It is recommended that the following identifiers not be used as flag name values:

- Names of Oracle parameters (for example, NLS_LENGTH_SEMANTICS)
- Identifiers with any of the following prefixes: PLS_, PLSQL_, PLSCC_, ORA_, ORACLE_, DBMS_, SYS_

**See Also:**

*Oracle Database PL/SQL Language Reference* for more information about this parameter

# PLSQL_CODE_TYPE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PLSQL_CODE_TYPE = { INTERPRETED \| NATIVE } |
| **Default value** | INTERPRETED |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

PLSQL_CODE_TYPE specifies the compilation mode for PL/SQL library units.

### Values

- INTERPRETED

  PL/SQL library units will be compiled to PL/SQL bytecode format. Such modules are executed by the PL/SQL interpreter engine.

- NATIVE

  PL/SQL library units (with the possible exception of top-level anonymous PL/SQL blocks) will be compiled to native (machine) code. Such modules will be executed natively without incurring any interpreter overhead.

When the value of this parameter is changed, it has no effect on PL/SQL library units that have already been compiled. The value of this parameter is stored persistently with each library unit.

If a PL/SQL library unit is compiled native, all subsequent automatic recompilations of that library unit will use native compilation.

**See Also:**

*Oracle Database PL/SQL Language Reference* for more information about this parameter

# PLSQL_DEBUG

| Property | Description |
|---|---|
| **Parameter type** | Boolean |

| Property | Description |
|---|---|
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

**Note:**

The `PLSQL_DEBUG` parameter is deprecated. It is retained for backward compatibility only.

`PLSQL_DEBUG` specifies whether or not PL/SQL library units will be compiled for debugging.

### Values

- `true`

  PL/SQL library units will be compiled for debugging

- `false`

  PL/SQL library units will be compiled for normal execution

When `PLSQL_DEBUG` is set to `true`, PL/SQL library units are always compiled `INTERPRETED` in order to be debuggable.

When the value of this parameter is changed, it has no effect on PL/SQL library units that have already been compiled. The value of this parameter is stored persistently with each library unit.

## PLSQL_OPTIMIZE_LEVEL

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `0` to `3` |
| **Basic** | No |

`PLSQL_OPTIMIZE_LEVEL` specifies the optimization level that will be used to compile PL/SQL library units. The higher the setting of this parameter, the more effort the compiler makes to optimize PL/SQL library units.

### Values

- `0`

Maintains the evaluation order and hence the pattern of side effects, exceptions, and package initializations of Oracle9*i* and earlier releases. Also removes the new semantic identity of `BINARY_INTEGER` and `PLS_INTEGER` and restores the earlier rules for the evaluation of integer expressions. Although code will run somewhat faster than it did in Oracle9*i*, use of level 0 will forfeit most of the performance gains of PL/SQL in Oracle Database 10*g*.

- `1`

  Applies a wide range of optimizations to PL/SQL programs including the elimination of unnecessary computations and exceptions, but generally does not move source code out of its original source order.

- `2`

  Applies a wide range of modern optimization techniques beyond those of level 1 including changes which may move source code relatively far from its original location.

- 3

  Applies a wide range of optimization techniques beyond those of level 2, automatically including techniques not specifically requested.

Generally, setting this parameter to `2` pays off in better execution performance. If, however, the compiler runs slowly on a particular source module or if optimization does not make sense for some reason (for example, during rapid turnaround development), then setting this parameter to `1` will result in almost as good a compilation with less use of compile-time resources.

The value of this parameter is stored persistently with the library unit.

**See Also:**

- *Oracle Database PL/SQL Language Reference* for more information about this parameter
- *Oracle Database Development Guide* for an example of using this parameter

# PLSQL_V2_COMPATIBILITY

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

**Note:**

The `PLSQL_V2_COMPATIBILITY` parameter is deprecated. It is retained for backward compatibility only.

PL/SQL Version 2 allows some abnormal behavior that Version 8 disallows. If you want to retain that behavior for backward compatibility, set PLSQL_V2_COMPATIBILITY to true. If you set it to false, then PL/SQL Version 8 behavior is enforced and Version 2 behavior is not allowed.

**See Also:**

*Oracle Database PL/SQL Language Reference* for a description of the differences between PL/SQL Version 2 and Version 8, and for more information on setting this parameter

# PLSQL_WARNINGS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | PLSQL_WARNINGS = '*value_clause*' [, '*value_clause*' ] ...<br>**value_clause::=**<br>{ ENABLE \| DISABLE \| ERROR }:<br>{ ALL<br>\| SEVERE<br>\| INFORMATIONAL<br>\| PERFORMANCE<br>\| { *integer*<br>\| (*integer* [, *integer* ] ...)<br>}<br>} |
| **Default value** | 'DISABLE:ALL' |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

**Examples**

```
PLSQL_WARNINGS = 'ENABLE:SEVERE', 'DISABLE:INFORMATIONAL';
PLSQL_WARNINGS = 'DISABLE:ALL';
PLSQL_WARNINGS = 'DISABLE:5000', 'ENABLE:5001', 'ERROR:5002';
PLSQL_WARNINGS = 'ENABLE:(5000,5001,5002)', 'DISABLE:(6000,6001)';
```

PLSQL_WARNINGS enables or disables the reporting of warning messages by the PL/SQL compiler, and specifies which warning messages to show as errors.

### value_clause

Multiple value clauses may be specified, enclosed in quotes and separated by commas. Each value clause is composed of a qualifier, a colon (:), and a modifier.

**Qualifier values:**

- ENABLE

  Enable a specific warning or a set of warnings

- DISABLE

  Disable a specific warning or a set of warnings

- ERROR

  Treat a specific warning or a set of warnings as errors

**Modifier values:**

- ALL

  Apply the qualifier to all warning messages

- SEVERE

  Apply the qualifier to only those warning messages in the SEVERE category

- INFORMATIONAL

  Apply the qualifier to only those warning messages in the INFORMATIONAL category

- PERFORMANCE

  Apply the qualifier to only those warning messages in the PERFORMANCE category

**See Also:**

*Oracle Database PL/SQL Language Reference* for more information about this parameter

# PRE_PAGE_SGA

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | No |
| **Range of values** | true \| false |
| **Basic** | No |

PRE_PAGE_SGA determines whether Oracle reads the entire SGA into memory at instance startup. Operating system page table entries are then prebuilt for each page of the SGA. This setting can increase the amount of time necessary for instance startup, but it is likely to decrease the amount of time necessary for Oracle to reach its full performance capacity after startup.

**Note:**

This setting does not prevent your operating system from paging or swapping the SGA after it is initially read into memory.

PRE_PAGE_SGA can increase the process startup duration, because every process that starts must access every page in the SGA. The cost of this strategy is fixed; however, you might simply determine that 20,000 pages must be touched every time a process starts. This approach can be useful with some applications, but not with all applications. Overhead can be significant if your system frequently creates and destroys processes by, for example, continually logging on and logging off.

The advantage that PRE_PAGE_SGA can afford depends on page size. For example, if the SGA is 80 MB and the page size is 4 KB, then 20,000 pages must be touched to refresh the SGA (80,000/4 = 20,000).

If the system permits you to set a 4 MB page size, then only 20 pages must be touched to refresh the SGA (80,000/4,000 = 20). The page size is operating system-specific and generally cannot be changed. Some operating systems, however, have a special implementation for shared memory whereby you can change the page size.

# PROCESSES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | This value is dynamic, and it depends on the number of CPUs. A computation factor of 80 is used for RDBMS instances and for Oracle ASM instances. |
| **Modifiable** | No |
| **Range of values** | 6 to operating system dependent |
| **Basic** | Yes |
| **Oracle RAC** | Multiple instances can have different values. |

PROCESSES specifies the maximum number of operating system user processes that can simultaneously connect to Oracle. Its value should allow for all background processes such as locks, job queue processes, and parallel execution processes.

The default values of the SESSIONS and TRANSACTIONS parameters are derived from this parameter. Therefore, if you change the value of PROCESSES, you should evaluate whether to adjust the values of those derived parameters.

**See Also:**

- Your operating system-specific Oracle documentation for the range of values
- *Oracle Database Concepts* for an introduction to Oracle database instance processes

# PROCESSOR_GROUP_NAME

| Property | Description |
|---|---|
| **Parameter type** | String |

| Property | Description |
|---|---|
| **Syntax** | PROCESSOR_GROUP_NAME = *string* |
| **Default value** | There is no default value |
| **Modifiable** | No |
| **Basic** | No |
| **Oracle RAC** | For Oracle RAC databases, it is recommended that the operating system processor groups for each database instance are all named the same, so that the same PROCESSOR_GROUP_NAME value can be used. However, you can choose to use different names. In this case, you would set the PROCESSOR_GROUP_NAME explicitly for each database instance. |

PROCESSOR_GROUP_NAME specifies the name of the processor group that this instance should run in.

PROCESSOR_GROUP_NAME instructs the database instance to run itself within the specified operating system processor group. All Oracle processes will be bound to the CPUs in this group and will only run on these CPUs. For NUMA systems, all SGA and PGA memory allocated by the database instance will be allocated from NUMA nodes within the group.

This initialization parameter is supported on the Linux and Solaris operating systems.

**Note:**

Oracle recommends that this parameter be set only for databases on which the USE_DEDICATED_BROKER initialization parameter is also set to TRUE.

**See Also:**

- "USE_DEDICATED_BROKER"
- *Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems* for more information about using the PROCESSOR_GROUP_NAME initialization parameter

# QUERY_REWRITE_ENABLED

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | QUERY_REWRITE_ENABLED = { false \| true \| force } |
| **Default value** | If OPTIMIZER_FEATURES_ENABLE is set to 10.0.0 or higher, then true<br>If OPTIMIZER_FEATURES_ENABLE is set to 9.2.0 or lower, then false |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |

| Property | Description |
|---|---|
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

Use QUERY_REWRITE_ENABLED to enable or disable query rewriting globally for the database.

### Values

- `false`

  Oracle does not use rewrite.

- `true`

  Oracle costs the query with rewrite and without rewrite and chooses the method with the lower cost.

- `force`

  Oracle always uses rewrite and does not evaluate the cost before doing so. Use `force` when you know that the query will always benefit from rewrite and when reduction in compile time is important.

To take advantage of query rewrite for a particular materialized view, you must enable query rewrite for that materialized view, and you must enable cost-based optimization.

**See Also:**

- *Oracle Database Data Warehousing Guide* for information on query rewrite of materialized views
- *Oracle Database SQL Tuning Guide* and "OPTIMIZER_MODE" for information on cost-based optimization

## QUERY_REWRITE_INTEGRITY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `QUERY_REWRITE_INTEGRITY = { enforced \| trusted \| stale_tolerated }` |
| **Default value** | `enforced` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`QUERY_REWRITE_INTEGRITY` determines the degree to which Oracle must enforce query rewriting. At the safest level, Oracle does not use query rewrite transformations that rely on unenforced relationships.

QUERY_REWRITE_INTEGRITY is relevant for materialized views as well as for foreign key constraints in NOVALIDATE state.

### Values

- enforced

  Oracle enforces and guarantees consistency and integrity.

- trusted

  Oracle allows rewrites using relationships that have been declared, but that are not enforced by Oracle.

- stale_tolerated

  Oracle allows rewrites using unenforced relationships. Materialized views are eligible for rewrite even if they are known to be inconsistent with the underlying detail data.

If a foreign key constraint is in NOVALIDATE state, join elimination is not done when QUERY_REWRITE_INTEGRITY=enforced. This means that queries with joins over a foreign key constraint that is in RELY NOVALIDATE state can potentially take longer to parse and execute as the optimizer does not trust the RELY.

**See Also:**

- *Oracle Database Data Warehousing Guide* for more information about query rewrite for materialized views
- *Oracle Database Data Warehousing Guide* for more information about the QUERY_REWRITE_INTEGRITY parameter

## RDBMS_SERVER_DN

| Property | Description |
|---|---|
| **Parameter type** | X.500 Distinguished Name |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | All X.500 Distinguished Name format values |
| **Basic** | No |

**Note:**

The RDBMS_SERVER_DN parameter is deprecated in Oracle Database 12*c* Release 1 (12.1.0.2). It is replaced by the LDAP_DIRECTORY_ACCESS initialization parameter.

RDBMS_SERVER_DN specifies the Distinguished Name (DN) of the Oracle server. It is used for retrieving Enterprise Roles from an enterprise directory service.

If you do not want to use a directory for enterprise user and privilege management, but prefer to use SSL authentication alone, do not set this parameter.

**See Also:**

- *Oracle Database Enterprise User Security Administrator's Guide* for more information on enterprise roles and the enterprise directory service
- "LDAP_DIRECTORY_ACCESS"

# READ_ONLY_OPEN_DELAYED

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`READ_ONLY_OPEN_DELAYED` determines when datafiles in read-only tablespaces are accessed.

### Values

- `true`

  The datafiles are accessed for the first time only when an attempt is made to read data stored within them.

- `false`

  The datafiles are accessed at database open time.

You can use this parameter to speed up some operations (primarily opening the database) for very large databases when substantial portions of the database are stored in read-only tablespaces. Consider setting this parameter to `true` for such databases, especially if portions of the read-only data are stored on slow-access devices or hierarchical storage.

**See Also:**

*Oracle Database Administrator's Guide* for information on the consequences of delaying access of datafiles in read-only tablespaces

# RECOVERY_PARALLELISM

| Property | Description |
|---|---|
| **Parameter type** | Integer |

| Property | Description |
|---|---|
| **Default value** | System-determined parallel recovery |
| **Modifiable** | No |
| **Range of values** | System-determined, but cannot exceed PARALLEL_MAX_SERVERS initialization parameter setting |
| **Basic** | No |

RECOVERY_PARALLELISM specifies the number of processes to participate in instance or crash recovery. To force serial crash and instance recovery, set the RECOVERY_PARALLELISM parameter to 0 or 1. 0 or 1 disable parallel instance and crash recovery on a system that has multiple CPUs. An alert log displays the degree of parallelism that was chosen when the database instance/recovery starts.

**See Also:**

- *Oracle Real Application Clusters Administration and Deployment Guide* for information on setting this parameter in an Oracle RAC environment

# RECYCLEBIN

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | RECYCLEBIN = { on \| off } |
| **Default value** | on |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM ... DEFERRED |
| **Basic** | No |

RECYCLEBIN is used to control whether the Flashback Drop capability is turned on or off. If the parameter is set to off, then dropped tables do not go into the recycle bin. If this parameter is set to on, then dropped tables go into the recycle bin and can be recovered.

**See Also:**

*Oracle Database Administrator's Guide* for more information about using Flashback Drop and managing the recycle bin

# REDO_TRANSPORT_USER

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | REDO_TRANSPORT_USER = *user_name* |

| Property | Description |
|---|---|
| **Default value** | There is no default value. |
| **Range of values** | Any character string that matches the name of a user who has been granted the SYSDBA or SYSOPER privilege |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |
| **Oracle RAC** | Every instance should use the same value |

REDO_TRANSPORT_USER specifies the name of the user whose password verifier is used when a remote login password file is used for redo transport authentication. This user must have the SYSDBA or SYSOPER privilege and must have the same password in the database that initiates the redo transport session and in the database that is the target of the redo transport session.

The value of this parameter is case sensitive and must exactly match the value of the USERNAME column of a row in the V$PWFILE_USERS view. The value of the SYSDBA or SYSOPER column of the row must also be TRUE.

If this parameter is not specified, then the password verifier of the SYS user will be used when a remote login password file is used for redo transport authentication.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about this parameter

# REMOTE_DEPENDENCIES_MODE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | REMOTE_DEPENDENCIES_MODE = { TIMESTAMP \| SIGNATURE } |
| **Default value** | TIMESTAMP |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

REMOTE_DEPENDENCIES_MODE specifies how Oracle should handle dependencies upon remote PL/SQL stored procedures.

### Values

- TIMESTAMP

  The client running the procedure compares the timestamp recorded on the server-side procedure with the current timestamp of the local procedure and executes the procedure only if the timestamps match.

- SIGNATURE

Oracle allows the procedure to execute as long as the signatures are considered safe. This setting allows client PL/SQL applications to be run without recompilation.

**See Also:**

*Oracle Database Development Guide* for suggestions for managing dependencies

# REMOTE_LISTENER

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `REMOTE_LISTENER = network_name` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | Yes |

`REMOTE_LISTENER` specifies a network name that resolves to an address or address list of Oracle Net remote listeners (that is, listeners that are not running on the same system as this instance). The address or address list is specified in the `TNSNAMES.ORA` file or other address repository as configured for your system.

**See Also:**

- *Oracle Database Concepts* for more information about listener processes and dispatcher processes
- *Oracle Database Net Services Administrator's Guide* and your operating system-specific Oracle documentation for more information about specifying network addresses for the protocols on your system

# REMOTE_LOGIN_PASSWORDFILE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `REMOTE_LOGIN_PASSWORDFILE = { shared \| exclusive \| none }` |
| **Default value** | `exclusive` |
| **Modifiable** | No |
| **Basic** | Yes |
| **Oracle RAC** | Multiple instances must have the same value. |

`REMOTE_LOGIN_PASSWORDFILE` specifies whether Oracle checks for a password file.

**Values**

- `shared`

  One or more databases can use the password file. The password file can contain `SYS` and non-`SYS` users.

  When `REMOTE_LOGIN_PASSWORDFILE` is set to `shared`:

  – The `SYS` password cannot be changed. If you try, the password change operation fails with "ORA-28046: Password change for SYS disallowed."

  – The password of any user who has `SYS*` admin privileges (`SYSDBA`, `SYSOPER`, `SYSASM`, `SYSBACKUP`, `SYSDG`, `SYSKM`) cannot be changed. If you try, the password change operation fails with "ORA-01999: password file cannot be updated in SHARED mode."

  – Grants of `SYS*` admin privileges (`SYSDBA`, `SYSOPER`, `SYSASM`, `SYSBACKUP`, `SYSDG`, `SYSKM`) to individual users are not allowed. For example, `grant sysdba to scott` fails with "ORA-01999: password file cannot be updated in SHARED mode." Similarly, revoke of `SYS*` admin privileges fails.

  – If the password file does not exist, then the behavior is the same as setting `REMOTE_LOGIN_PASSWORDFILE` to `none`.

- `exclusive`

  The password file can be used by only one database. The password file can contain `SYS` and non-`SYS` users.

  When `REMOTE_LOGIN_PASSWORDFILE` is set to `exclusive`, if the password file does not exist, then the behavior is the same as setting `REMOTE_LOGIN_PASSWORDFILE` to `none`.

- `none`

  Oracle ignores any password file. Therefore, privileged users must be authenticated by the operating system.

When remote login is attempted (such as `sqlplus sys/`*`password_for_sys@tns_name_of_database`* `as sysdba`), then:

- If a password file does not exist, error "ORA-01031: insufficient privileges" will be raised.
- If the password is incorrect, error "ORA-01017: invalid username/password; logon denied" will be raised.
- If the username is incorrect, error "ORA-01031: insufficient privileges" will be raised.

---

**Note:**

If you change `REMOTE_LOGIN_PASSWORDFILE` to `exclusive` or `shared` from `none`, then ensure that the password file is synchronized with the dictionary passwords. See *Oracle Database Administrator's Guide* for more information.

---

# REMOTE_OS_AUTHENT

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`REMOTE_OS_AUTHENT` specifies whether remote clients will be authenticated with the value of the `OS_AUTHENT_PREFIX` parameter.

**Note:**

The `REMOTE_OS_AUTHENT` parameter is deprecated. It is retained for backward compatibility only.

**See Also:**

"OS_AUTHENT_PREFIX"

# REMOTE_OS_ROLES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`REMOTE_OS_ROLES` specifies whether operating system roles are allowed for remote clients. The default value, `false`, causes Oracle to identify and manage roles for remote clients.

**See Also:**

- *Oracle Database Security Guide* for more information on setting this parameter
- "OS_ROLES"

# REPLICATION_DEPENDENCY_TRACKING

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`REPLICATION_DEPENDENCY_TRACKING` enables or disables dependency tracking for read/write operations to the database. Dependency tracking is essential for propagating changes in a replicated environment in parallel.

### Values

- `TRUE`

  Enables dependency tracking.

- `FALSE`

  Allows read/write operations to the database to run faster, but does not produce dependency information for Oracle to perform parallel propagation.

---

**Note:**

Do not specify this value unless you are sure that your application will not perform any read/write operations to the replicated tables.

---

**See Also:**

*Oracle Database Advanced Replication* for more information on parallel propagation dependency tracking

---

# RESOURCE_LIMIT

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

`RESOURCE_LIMIT` determines whether resource limits are enforced in database profiles.

### Values

- TRUE

  Enables the enforcement of resource limits

- FALSE

  Disables the enforcement of resource limits

**See Also:**

*Oracle Database Administrator's Guide* and *Oracle Database SQL Language Reference* for more information on setting resource limits for profiles

## RESOURCE_MANAGER_CPU_ALLOCATION

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | The number of logical CPUs reported by the operating system. |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 to operating system-specific |
| **Basic** | No |

**Note:**

The RESOURCE_MANAGER_CPU_ALLOCATION parameter is deprecated. It is retained for backward compatibility only. Use the CPU_COUNT parameter instead.

RESOURCE_MANAGER_CPU_ALLOCATION specifies the number of CPUs that the Resource Manager should use. The Resource Manager controls how a system's CPUs are utilized by its database's sessions.

The Resource Manager schedules database sessions on the CPUs according to a resource plan that has been configured and enabled by the DBA. Normally, the Resource Manager schedules enough database sessions to keep all CPUs utilized. However, in some scenarios, a DBA may only want to schedule enough database sessions to keep a subset of the CPUs utilized.

**See Also:**

"CPU_COUNT"

## RESOURCE_MANAGER_PLAN

| Property | Description |
|---|---|
| **Parameter type** | String |

| Property | Description |
|---|---|
| **Syntax** | `RESOURCE_MANAGER_PLAN = resource_plan_name` |
| **Default value** | There is no default value. |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Any valid character string |
| **Basic** | No |

### In a CDB

In the root for a CDB, `RESOURCE_MANAGER_PLAN` specifies the CDB resource plan. A CDB resource plan allocates resources among PDBs.

A CDB resource plan is created using `DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN` and `CREATE_CDB_PLAN_DIRECTIVE`.

**See Also:**

- *Oracle Database Administrator's Guide* for more information about using `DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN` and `DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE`
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `CREATE_CDB_PLAN` procedure and `CREATE_CDB_PLAN_DIRECTIVE` procedure for the `DBMS_RESOURCE_MANAGER` package

A session must be `root` to change the value of `RESOURCE_MANAGER_PLAN` for a CDB using the `ALTER SYSTEM` statement. For example, to enable and disable a CDB resource plan:

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = CDB_resource_plan_name;
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = '';
```

In a PDB, `RESOURCE_MANAGER_PLAN` specifies the PDB resource plan to use for the PDB.

A session must be in the PDB to enable or disable a PDB resource plan for that PDB. For example, to enable and disable a PDB resource plan:

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = PDB_resource_plan_name;
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = '';
```

In a PDB, the PDB resource plan has some restrictions compared to the resource plan of a non-CDB. The following restrictions apply to PDB resource plans:

- A PDB resource plan cannot have subplans.
- A PDB resource plan can have a maximum of eight consumer groups.
- A PDB resource plan cannot have a multiple-level scheduling policy.

To enforce certain PDB resource plan policies, policies regarding resource allocation among PDBs should exist in the CDB resource plan. Without a CDB resource plan, certain PDB resource plan policies will not be enforced. If a PDB resource plan contains CPU or parallel statement queuing directives and a CDB resource plan is not specified, then Resource Manager will automatically enable the `DEFAULT_CDB_PLAN` plan. To prevent this behavior, set the `RESOURCE_MANAGER_PLAN` parameter at the root level to `ORA$INTERNAL_CDB_PLAN`.

**Note:**

See *Oracle Database Administrator's Guide* for information on CPU, I/O bandwidth, and parallel execution servers requirements in CDB resource plans, and for a description of the results in the PDB resource plans when those requirements are not met.

### In a Non-CDB

`RESOURCE_MANAGER_PLAN` specifies the top-level resource plan to use for an instance in a non-CDB. The resource manager will load this top-level resource plan along with all its descendants (subplans, directives, and consumer groups). If you do not specify this parameter, the resource manager is off by default.

You can change the setting of this parameter using the `ALTER SYSTEM` statement to turn on the resource manager (if it was previously off) or to turn off the resource manager or change the current resource plan (if it was previously on). If you specify a resource plan that does not exist in the data dictionary, Oracle returns an error message.

**See Also:**

- *Oracle Database Administrator's Guide* for information on resource plans
- *Oracle Database Administrator's Guide* for information on using Oracle Resource Manager with a CDB and PDBs
- *Oracle Database PL/SQL Packages and Types Reference* for information on the `DBMS_RESOURCE_MANAGER` package
- *Oracle Database PL/SQL Packages and Types Reference* for information on the `DBMS_RESOURCE_MANAGER_PRIVS` package
- "DBA_RSRC_PLANS", "DBA_RSRC_PLAN_DIRECTIVES", and the various `V$RSRC_*` dynamic performance views in Dynamic Performance Views for information on existing resource plans

# RESULT_CACHE_MAX_RESULT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 5 percent |
| **Modifiable** | `ALTER SYSTEM` |

| Property | Description |
|---|---|
| **Range of values** | 0 to 100 |
| **Basic** | No |

RESULT_CACHE_MAX_RESULT specifies the percentage of RESULT_CACHE_MAX_SIZE that any single result can use.

# RESULT_CACHE_MAX_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | RESULT_CACHE_MAX_SIZE = *integer* [K \| M \| G] |
| **Default value** | Derived from the values of SHARED_POOL_SIZE, SGA_TARGET, and MEMORY_TARGET |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 0 to operating system-dependent |
| **Basic** | No |
| **Oracle RAC** | You must either set this parameter to 0 on all instances to disable the result cache, or use a nonzero value on all instances. Disabling the result cache on some instances may lead to incorrect results. |

RESULT_CACHE_MAX_SIZE specifies the maximum amount of SGA memory (in bytes) that can be used by the Result Cache. Values of this parameter greater than 0 are rounded up to the next multiple of 32 KB. If the value of this parameter is 0, then the feature is disabled.

**See Also:**

*Oracle Database Performance Tuning Guide* for information about tuning the result cache

# RESULT_CACHE_MODE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | RESULT_CACHE_MODE = { MANUAL \| FORCE } |
| **Default value** | MANUAL |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

RESULT_CACHE_MODE specifies when a ResultCache operator is spliced into a query's execution plan.

### Values

- MANUAL

  The ResultCache operator is added only when the query is annotated (that is, hints).

- FORCE

  The ResultCache operator is added to the root of all SELECT statements (when it is valid to do so).

---

**Note:**

FORCE mode is not recommended because the database and clients will attempt to cache all queries, which may create significant performance and latching overhead. Moreover, because queries that call non-deterministic PL/SQL functions are also cached, enabling the result cache in such a broad-based manner may cause material changes to the results.

---

For the FORCE setting, if the statement contains a NO_RESULT_CACHE hint, then the hint takes precedence over the parameter setting.

---

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information on how the result cache handles PL/SQL functions before changing the value of this initialization parameter
- *Oracle Database SQL Language Reference* for more information about the NO_RESULT_CACHE hint

---

## RESULT_CACHE_REMOTE_EXPIRATION

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | 0 to operating system-dependent |
| **Basic** | No |

RESULT_CACHE_REMOTE_EXPIRATION specifies the number of minutes that a result using a remote object is allowed to remain valid. Setting this parameter to 0 implies that results using remote objects should not be cached. Setting this parameter to a nonzero value can produce stale answers (for example, if the remote table used by a result is modified at the remote database).

**See Also:**

*Oracle Database Performance Tuning Guide* for information about tuning the result cache

## RESUMABLE_TIMEOUT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` (seconds) |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `0` to $2^{31}$ - 1 (in seconds) |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`RESUMABLE_TIMEOUT` enables or disables resumable statements and specifies resumable timeout at the system level.

**See Also:**

*Oracle Database Administrator's Guide* for more information about enabling resumable space allocation, what conditions are correctable, and what statements can be made resumable

## ROLLBACK_SEGMENTS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `ROLLBACK_SEGMENTS =`<br>`(segment_name [, segment_name] ... )` |
| **Default value** | If you do not specify this parameter, the instance uses public rollback segments by default, unless the `UNDO_MANAGEMENT` initialization parameter is set to `AUTO`. In that case, the `ROLLBACK_SEGMENTS` parameter is ignored and automatic undo management is used. |
| **Modifiable** | No |
| **Range of values** | Any rollback segment names listed in `DBA_ROLLBACK_SEGS` except `SYSTEM` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must have different values. |

`ROLLBACK_SEGMENTS` allocates one or more rollback segments by name to this instance. If you set this parameter, the instance acquires all of the rollback segments named in this parameter, even if the number of rollback segments exceeds the

minimum number required by the instance (calculated as `TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT`).

You cannot change the value of this parameter dynamically, but you can change its value and then restart the instance. Although this parameter usually specifies private rollback segments, it can also specify public rollback segments if they are not already in use.

To find the name, segment ID number, and status of each rollback segment in the database, query the data dictionary view `DBA_ROLLBACK_SEGS`.

When `UNDO_MANAGEMENT` is set to `AUTO`, `ROLLBACK_SEGMENTS` is ignored.

**See Also:**

- *Oracle Real Application Clusters Administration and Deployment Guide* for information on setting this parameter in an Oracle RAC environment
- "DBA_ROLLBACK_SEGS"

# SEC_CASE_SENSITIVE_LOGON

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

**Note:**

The `SEC_CASE_SENSITIVE_LOGON` parameter is deprecated. It is retained for backward compatibility only. For more information about the deprecation of this parameter, see *Oracle Database Security Guide*.

`SEC_CASE_SENSITIVE_LOGON` enables or disables password case sensitivity in the database.

### Values

- `true`

  Database logon passwords are case sensitive.

- `false`

  Database logon passwords are not case sensitive.

**See Also:**

*Oracle Database Security Guide* for more details about the SEC_CASE_SENSITIVE_LOGON parameter

# SEC_MAX_FAILED_LOGIN_ATTEMPTS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 3 |
| **Modifiable** | No |
| **Range of values** | 1 or higher. In this context, "higher" means any integer value higher than 1, not the string "higher." |
| **Basic** | No |

SEC_MAX_FAILED_LOGIN_ATTEMPTS specifies the number of authentication attempts that can be made by a client on a connection to the server process. These login attempts can be for multiple user accounts in the same connection. After the specified number of failure attempts, the connection will be automatically dropped by the server process.

**See Also:**

*Oracle Database 2 Day + Security Guide* for more information about this parameter

# SEC_PROTOCOL_ERROR_FURTHER_ACTION

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | SEC_PROTOCOL_ERROR_FURTHER_ACTION = { CONTINUE \| (DELAY,*integer*) \| (DROP,*integer*) } |
| **Default value** | (DROP,3) |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

SEC_PROTOCOL_ERROR_FURTHER_ACTION specifies the further execution of a server process when receiving bad packets from a possibly malicious client.

### Values

- CONTINUE

  The server process continues execution. The database server may be subject to a Denial of Service (DoS) if bad packets continue to be sent by a malicious client.

- (DELAY,*integer*)

  The client experiences a delay of *integer* seconds before the server process accepts the next request from the same client connection. Malicious clients are prevented from excessive consumption of server resources while legitimate clients experience a degradation in performance but can continue to function.

- (DROP,*integer*)

  The server forcefully terminates the client connection after *integer* cumulative bad packets. The server protects itself at the expense of the client (for example, a client transaction may be lost). The client may reconnect and attempt the same operation.

**See Also:**

*Oracle Database Security Guide* for examples of using this parameter

# SEC_PROTOCOL_ERROR_TRACE_ACTION

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | SEC_PROTOCOL_ERROR_TRACE_ACTION = { NONE \| TRACE \| LOG \| ALERT } |
| **Default value** | TRACE |
| **Modifiable** | ALTER SYSTEM |
| **Basic** | No |

SEC_PROTOCOL_ERROR_TRACE_ACTION specifies the action that the database should take when bad packets are received from a possibly malicious client.

### Values

- NONE

  The database server ignores the bad packets and does not generate any trace files or log messages.

- TRACE

  A detailed trace file is generated when bad packets are received, which can be used to debug any problems in client/server communication.

- LOG

  A minimal log message is printed in the alert logfile and in the server trace file. A minimal amount of disk space is used.

- ALERT

  An alert message is sent to a DBA or monitoring console.

# SEC_RETURN_SERVER_RELEASE_BANNER

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

SEC_RETURN_SERVER_RELEASE_BANNER specifies whether the server returns complete database software information to unauthenticated clients.

### Values

- `true`

  Returns complete database version information to the client.

- `false`

  Returns a generic version string to the client.

**See Also:**

*Oracle Call Interface Programmer's Guide* and *Oracle Database Security Guide* for more information on controlling the display of the database version banner.

## SERIAL_REUSE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `SERIAL_REUSE = { disable \| all \| select \| dml \| plsql \| force }` |
| **Default value** | `disable` |
| **Modifiable** | No |
| **Basic** | No |

**Note:**

The SERIAL_REUSE parameter is deprecated. It is retained for backward compatibility only.

SERIAL_REUSE specifies which types of cursors make use of the serial-reusable memory feature. This feature allocates private cursor memory in the SGA so that it can be reused (serially, not concurrently) by sessions executing the same cursor.

### Values

- `disable`

  Disables the option for all SQL statement types. This value overrides any other values included in the list.

- `all`

  Enables the option for both DML and `SELECT` statements. Equivalent to setting `select`, `dml`, and `plsql`.

- `select`

  Enables the option for `SELECT` statements.

- `dml`

  Enables the option for DML statements.

- `plsql`

  Currently has no effect (although PL/SQL packages do support the serial-reuse memory option using PL/SQL pragmas).

---

**Note:**

If `CURSOR_SPACE_FOR_TIME` is set to `true`, then the value of `SERIAL_REUSE` is ignored and treated as if it were set to `disable`.

---

---

**See Also:**

"CURSOR_SPACE_FOR_TIME"

---

## SERVICE_NAMES

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `SERVICE_NAMES =`<br>*`db_service_name`* `[,` *`db_service_name`* `[ ... ] ]` |
| **Default value** | `DB_UNIQUE_NAME.DB_DOMAIN` if defined |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Any ASCII string or comma-separated list of string names |
| **Basic** | No |
| **Oracle RAC** | Do not set the `SERVICE_NAMES` parameter for Oracle RAC environments. Instead, define services using Oracle Enterprise Manager and manage those services using Server Control (SRVCTL) utility. |

`SERVICE_NAMES` specifies one or more names by which clients can connect to the instance. The instance registers its service names with the listener. When a client

requests a service, the listener determines which instances offer the requested service and routes the client to the appropriate instance.

You can specify multiple service names to distinguish among different uses of the same database. For example:

```
SERVICE_NAMES = sales.example.com, widgetsales.example.com
```

You can also use service names to identify a single service that is available from two different databases through the use of replication.

If you do not qualify the names in this parameter with a domain, Oracle qualifies them with the value of the `DB_DOMAIN` parameter. If `DB_DOMAIN` is not specified, then no domain will be applied to the non-qualified `SERVICE_NAMES` values.

When you specify additional service names with this parameter, the default service name is not overridden. The default service name plus the additional service names specified with this parameter are the service names that clients can use to connect to the database.

**See Also:**

- *Oracle Database Net Services Administrator's Guide* for more information on this parameter and its settings
- *Oracle Real Application Clusters Administration and Deployment Guide* for information about services administration in an Oracle RAC environment
- "DB_DOMAIN"

# SESSION_CACHED_CURSORS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `50` |
| **Modifiable** | `ALTER SESSION` |
| **Range of values** | 0 to operating system-dependent |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

`SESSION_CACHED_CURSORS` specifies the number of session cursors to cache. Repeated parse calls of the same SQL (including recursive SQL) or PL/SQL statement cause the session cursor for that statement to be moved into the session cursor cache. Oracle uses a least recently used algorithm to remove entries in the session cursor cache to make room for new entries when needed.

**See Also:**

*Oracle Database Performance Tuning Guide* for information about enabling the session cursor cache

# SESSION_MAX_OPEN_FILES

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `10` |
| **Modifiable** | No |
| **Range of values** | 1 to either 50 or the value of `MAX_OPEN_FILES` defined at the operating system level, whichever is less |
| **Basic** | No |

`SESSION_MAX_OPEN_FILES` specifies the maximum number of BFILEs that can be opened in any session. Once this number is reached, subsequent attempts to open more files in the session by using `DBMS_LOB.FILEOPEN()` or `OCILobFileOpen()` will fail. The maximum value for this parameter depends on the equivalent parameter defined for the underlying operating system.

**See Also:**

- *Oracle Database SecureFiles and Large Objects Developer's Guide* for information on LOBs in general
- *Oracle Database SecureFiles and Large Objects Developer's Guide* for information on BFILEs
- *Oracle Database PL/SQL Packages and Types Reference* for information on the `DBMS_LOB.FILEOPEN()` procedure
- *Oracle Call Interface Programmer's Guide* for information on the `OCILobFileOpen()` procedure

# SESSIONS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | Derived: (1.5 * `PROCESSES`) + 22 |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | 1 to $2^{16}$ (which is 1 to 65536) |
| **Basic** | Yes |

`SESSIONS` specifies the maximum number of sessions that can be created in the system. Because every login requires a session, this parameter effectively determines the maximum number of concurrent users in the system. You should always set this parameter explicitly to a value equivalent to your estimate of the maximum number of concurrent users, plus the number of background processes, plus approximately 10% for recursive sessions.

Oracle uses the default value of this parameter as its minimum. Values between 1 and the default do not trigger errors, but Oracle ignores them and uses the default instead.

The default values of the ENQUEUE_RESOURCES and TRANSACTIONS parameters are derived from SESSIONS. Therefore, if you increase the value of SESSIONS, you should consider whether to adjust the values of ENQUEUE_RESOURCES and TRANSACTIONS as well. (Note that ENQUEUE_RESOURCES is obsolete as of Oracle Database 10g release 2 (10.2).)

In a shared server environment, the value of PROCESSES can be quite small. Therefore, Oracle recommends that you adjust the value of SESSIONS to approximately 1.1 * total number of *connections*.

For a CDB, the root container's SESSIONS parameter specifies the total number of sessions for the database.

The SESSIONS parameter for a PDB specifies the total number of sessions that can be used by that PDB. Its value defaults to the root container's SESSIONS value. If the PDB tries to use more sessions than configured by its SESSIONS parameter, an ORA-00018 error message is generated.

The SESSIONS parameter for a PDB can only be modified by the PDB. It cannot be set higher than the CDB's SESSIONS value.

**See Also:**

- *Oracle Database Concepts* for more information on memory structures
- *Oracle Database Concepts* for more information on processes
- *Oracle Database Concepts* for more information about CDBs and PDBs

# SGA_MAX_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | SGA_MAX_SIZE = *integer* [K \| M \| G] |
| **Default value** | Initial size of SGA at startup, dependent on the sizes of different pools in the SGA, such as buffer cache, shared pool, large pool, and so on. |
| **Modifiable** | No |
| **Range of values** | 0 to operating system-dependent |

SGA_MAX_SIZE specifies the maximum size of the SGA for the lifetime of the instance.

On 64-bit platforms and non-Windows 32-bit platforms, when either MEMORY_TARGET or MEMORY_MAX_TARGET is specified, the default value of SGA_MAX_SIZE is set to the larger of the two parameters. This causes more address space to be reserved for expansion of the SGA.

On Windows 32-bit platforms, the default value of SGA_MAX_SIZE is the largest of the following values:

- 60% of MEMORY_TARGET, if specified

- 60% of MEMORY_MAX_TARGET, if specified
- 25% of the total available virtual address space

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this parameter

# SGA_TARGET

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | SGA_TARGET = *integer* [K \| M \| G] |
| **Default value** | 0 (SGA autotuning is disabled for DEFERRED mode autotuning requests, but allowed for IMMEDIATE mode autotuning requests) |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | 64 MB to operating system-dependent |
| **Basic** | Yes |

SGA_TARGET specifies the total size of all SGA components. If SGA_TARGET is specified, then the following memory pools are automatically sized:

- Buffer cache (DB_CACHE_SIZE)
- Shared pool (SHARED_POOL_SIZE)
- Large pool (LARGE_POOL_SIZE)
- Java pool (JAVA_POOL_SIZE)
- Streams pool (STREAMS_POOL_SIZE)

If these automatically tuned memory pools are set to nonzero values, then those values are used as minimum levels by Automatic Shared Memory Management. You would set minimum values if an application component needs a minimum amount of memory to function properly.

The following pools are manually sized components and are not affected by Automatic Shared Memory Management:

- Log buffer
- Other buffer caches, such as KEEP, RECYCLE, and other block sizes
- Fixed SGA and other internal allocations

The memory allocated to these pools is deducted from the total available for SGA_TARGET when Automatic Shared Memory Management computes the values of the automatically tuned memory pools.

In the **Default value** field, IMMEDIATE mode autotuning requests are necessary to avoid ORA-04031 errors. The DEFERRED and IMMEDIATE modes are reflected in the OPER_MODE column of the V$MEMORY_RESIZE_OPS view.

If Automatic Memory Management is enabled (MEMORY_TARGET is set to a positive value) and SGA_TARGET is also set to a positive value, the SGA_TARGET value acts as the minimum value for the size of the SGA.

**See Also:**

- "DB_CACHE_SIZE"
- "SHARED_POOL_SIZE"
- "LARGE_POOL_SIZE"
- "JAVA_POOL_SIZE"
- "STREAMS_POOL_SIZE"
- "V$MEMORY_RESIZE_OPS"
- "MEMORY_TARGET"
- "PGA_AGGREGATE_TARGET"
- *Oracle Database Administrator's Guide* for information on automatic memory management
- *Oracle Database Administrator's Guide* for information on managing the SGA manually

# SHADOW_CORE_DUMP

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | SHADOW_CORE_DUMP = { partial \| full \| none } |
| **Default value** | partial |
| **Modifiable** | No |
| **Basic** | No |

SHADOW_CORE_DUMP specifies whether Oracle includes the SGA in the core file for foreground (client) processes.

### Values

- partial

  Oracle does not include the SGA in the core dump.

- full

  Oracle includes the SGA in the core dump.

- none

  No core files will be generated for foreground processes.

**See Also:**

"BACKGROUND_CORE_DUMP"

# SHARED_MEMORY_ADDRESS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 0 |
| **Modifiable** | No |
| **Basic** | No |

SHARED_MEMORY_ADDRESS and HI_SHARED_MEMORY_ADDRESS specify the starting address at run time of the system global area (SGA). This parameter is ignored on the many platforms that specify the SGA's starting address at linktime.

Use this parameter to specify the entire address on 32-bit platforms and to specify the low-order 32 bits of a 64-bit address on 64-bit platforms. Use HI_SHARED_MEMORY_ADDRESS to specify the high-order 32 bits of a 64-bit address on 64-bit platforms. If both parameters are 0 or unspecified, the SGA address defaults to a platform-specific location.

**See Also:**

"HI_SHARED_MEMORY_ADDRESS"

# SHARED_POOL_RESERVED_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | SHARED_POOL_RESERVED_SIZE = *integer* [K \| M \| G] |
| **Default value** | 5% of the value of SHARED_POOL_SIZE |
| **Modifiable** | No |
| **Range of values** | Minimum: 5000<br>Maximum: half of the value of SHARED_POOL_SIZE |
| **Basic** | No |

SHARED_POOL_RESERVED_SIZE specifies (in bytes) the shared pool space that is reserved for large contiguous requests for shared pool memory.

You can use this parameter to avoid performance degradation in the shared pool in situations where pool fragmentation forces Oracle to search for and free chunks of unused pool to satisfy the current request.

**See Also:**

- "SHARED_POOL_SIZE"
- *Oracle Database Performance Tuning Guide* for information on sizing the shared pool

# SHARED_POOL_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `SHARED_POOL_SIZE = integer [K \| M \| G]` |
| **Default value** | If `SGA_TARGET` is set: If the parameter is not specified, then the default is `0` (internally determined by the Oracle Database). If the parameter is specified, then the user-specified value indicates a minimum value for the memory pool.<br>If `SGA_TARGET` is not set (32-bit platforms): `64` MB, rounded up to the nearest granule size.<br>If `SGA_TARGET` is not set (64-bit platforms): `128` MB, rounded up to the nearest granule size.<br>For considerations when dealing with database instances using Oracle ASM, see "SHARED_POOL_SIZE and Automatic Storage Management". |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Minimum: the granule size<br>Maximum: operating system-dependent |
| **Basic** | No |

`SHARED_POOL_SIZE` specifies (in bytes) the size of the shared pool. The shared pool contains shared cursors, stored procedures, control structures, and other structures. If you set `PARALLEL_AUTOMATIC_TUNING` to `false`, then Oracle also allocates parallel execution message buffers from the shared pool. Larger values improve performance in multiuser systems. Smaller values use less memory.

You can monitor utilization of the shared pool by querying the view `V$SGASTAT`.

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information on setting this parameter
- *Oracle Database Upgrade Guide* for information on parallel execution message buffers
- "PARALLEL_AUTOMATIC_TUNING" and "V$SGASTAT"

### SHARED_POOL_SIZE and Automatic Storage Management

On a database instance using Oracle Automatic Storage Management (Oracle ASM), additional memory is required to store extent maps. As a general guideline, you can aggregate the values from the following queries to obtain current database storage size that is either already on Oracle ASM or will be stored in Oracle ASM. Then determine the redundancy type that is used (or will be used), and calculate the value for SHARED_POOL_SIZE, using the aggregated value as input.

```
SELECT SUM(BYTES)/(1024*1024*1024) FROM V$DATAFILE;
SELECT SUM(BYTES)/(1024*1024*1024) FROM V$LOGFILE a, V$LOG b
WHERE a.group#=b.group#;
SELECT SUM(BYTES)/(1024*1024*1024) FROM V$TEMPFILE WHERE
status='ONLINE';
```

Additionally, keep the following guidelines in mind:

- For diskgroups using external redundancy:

  (Every 100G of space needs 1M of extra shared pool) + 2M

- For diskgroups using normal redundancy:

  (Every 50G of space needs 1M of extra shared pool) + 4M

- For diskgroups using high redundancy:

  (Every 33G of space needs 1M of extra shared pool) + 6M

## SHARED_SERVER_SESSIONS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | If SHARED_SERVER_SESSIONS is specified, then it should be less than SESSIONS. If SHARED_SERVER_SESSIONS is not specified, then a shared server session may be created as long as there is a free session slot. |
| **Basic** | No |

SHARED_SERVER_SESSIONS specifies the total number of shared server sessions to allow. Setting this parameter enables you to reserve user sessions for dedicated servers.

**See Also:**

*Oracle Database Concepts* for more information on sessions

## SHARED_SERVERS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0`, meaning that shared server is not on.<br>If you are using shared server architecture or if the `DISPATCHERS` parameter is set such that the total number of dispatchers is more than 0, then the default value is `1`. |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | The value of this parameter should be less than `MAX_SHARED_SERVERS`. If it is greater than or equal to `MAX_SHARED_SERVERS`, then the number of servers will not be self-tuned but will remain constant, as specified by `SHARED_SERVERS`. |
| **Basic** | Yes |

`SHARED_SERVERS` specifies the number of server processes that you want to create when an instance is started. If system load decreases, then this minimum number of servers is maintained. Therefore, you should take care not to set `SHARED_SERVERS` too high at system startup.

Starting with Oracle Database 12*c* Release 1 (12.1.0.2), the `SHARED_SERVERS` parameter can be set on PDBs. However, unlike most other parameters that can be set on a PDB, this parameter can only be used within a PDB to enable or disable use of shared servers for that PDB. Therefore, in a PDB, the DBA can either set `SHARED_SERVERS` to 0 to disable use of shared servers for that PDB or use `ALTER SYSTEM RESET SHARED_SERVERS` to re-enable shared servers for the PDB. The configuration of shared servers for the CDB can only be done in the root.

After using `ALTER SYSTEM RESET SHARED SERVERS` to re-enable shared servers for a PDB, you must restart the PDB for shared servers to be re-enabled.

**See Also:**

- "DISPATCHERS"
- "MAX_SHARED_SERVERS"
- *Oracle Database Administrator's Guide* for more information on setting this parameter

# SKIP_UNUSABLE_INDEXES

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |

SKIP_UNUSABLE_INDEXES enables or disables the use and reporting of tables with unusable indexes or index partitions. If a SQL statement uses a hint that forces the usage of an unusable index, then this hint takes precedence over initialization parameter settings, including SKIP_UNUSABLE_INDEXES. If the optimizer chooses an unusable index, then an ORA-01502 error will result. (See *Oracle Database Administrator's Guide* for more information about using hints.)

### Values

- `true`

  Disables error reporting of indexes and index partitions marked UNUSABLE. This setting allows all operations (inserts, deletes, updates, and selects) on tables with unusable indexes or index partitions.

  **Note:**

  If an index is used to enforce a UNIQUE constraint on a table, then allowing insert and update operations on the table might violate the constraint. Therefore, this setting does not disable error reporting for unusable indexes that are unique.

- `false`

  Enables error reporting of indexes marked UNUSABLE. This setting does not allow inserts, deletes, and updates on tables with unusable indexes or index partitions.

  **See Also:**

  *Oracle Database SQL Language Reference* for more information about hints

## SMTP_OUT_SERVER

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | SMTP_OUT_SERVER = *server_clause* [, *server_clause* ] ...<br>**server_clause::=**<br>*host_name*[:*port*] |
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

SMTP_OUT_SERVER specifies the SMTP host and port to which UTL_MAIL delivers out-bound E-mail. Multiple servers may be specified, separated by commas.

If the first server in the list is unavailable, then UTL_MAIL tries the second server, and so on.

If SMTP_OUT_SERVER is not specified, then the SMTP server name defaults to the value of DB_DOMAIN, the port number defaults to 25, and the SMTP domain defaults to the suffix of DB_DOMAIN.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for information on the UTL_MAIL package

# SORT_AREA_RETAINED_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | Derived from SORT_AREA_SIZE |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM ... DEFERRED |
| **Range of values** | From the value equivalent of two database blocks to the value of SORT_AREA_SIZE |
| **Basic** | No |

**Note:**

Oracle does not recommend using the SORT_AREA_RETAINED_SIZE parameter unless the instance is configured with the shared server option. Oracle recommends that you enable automatic sizing of SQL working areas by setting PGA_AGGREGATE_TARGET instead. SORT_AREA_RETAINED_SIZE is retained for backward compatibility.

SORT_AREA_RETAINED_SIZE specifies (in bytes) the maximum amount of the user global area (UGA) memory retained after a sort run completes. The retained size controls the size of the read buffer, which Oracle uses to maintain a portion of the sort in memory. This memory is released back to the UGA, not to the operating system, after the last row is fetched from the sort space.

Oracle may allocate multiple sort spaces of this size for each query. Usually, only one or two sorts occur simultaneously, even for complex queries. In some cases, however, additional concurrent sorts are required, and each sort keeps its own memory area. If the shared server is used, allocation is to the SGA until the value in SORT_AREA_RETAINED_SIZE is reached. The difference between SORT_AREA_RETAINED_SIZE and SORT_AREA_SIZE is allocated to the PGA.

**Note:**

The default value as reflected in the V$PARAMETER dynamic performance view is 0. However, if you do not explicitly set this parameter, Oracle actually uses the value of the SORT_AREA_SIZE parameter.

**See Also:**

- "SORT_AREA_SIZE"
- *Oracle Database Performance Tuning Guide* for information on setting the values of this parameter and the SORT_AREA_SIZE parameter to tune sort operations using shared servers

# SORT_AREA_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 65536 |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM ... DEFERRED |
| **Range of values** | Minimum: the value equivalent of six database blocks<br>Maximum: operating system-dependent |
| **Basic** | No |

**Note:**

Oracle does not recommend using the SORT_AREA_SIZE parameter unless the instance is configured with the shared server option. Oracle recommends that you enable automatic sizing of SQL working areas by setting PGA_AGGREGATE_TARGET instead. SORT_AREA_SIZE is retained for backward compatibility.

SORT_AREA_SIZE specifies (in bytes) the maximum amount of memory Oracle will use for a sort. After the sort is complete, but before the rows are returned, Oracle releases all of the memory allocated for the sort, except the amount specified by the SORT_AREA_RETAINED_SIZE parameter. After the last row is returned, Oracle releases the remainder of the memory.

Increasing SORT_AREA_SIZE size improves the efficiency of large sorts.

Each sort in a query can consume memory up to the amount specified by SORT_AREA_SIZE, and there can be multiple sorts in a query. Also, if a query is executed in parallel, each PQ slave can consume memory up to the amount specified by SORT_AREA_SIZE for each sort it does.

SORT_AREA_SIZE is also used for inserts and updates to bitmap indexes. Setting this value appropriately results in a bitmap segment being updated only once for each DML operation, even if more than one row in that segment changes.

Larger values of SORT_AREA_SIZE permit more sorts to be performed in memory. If more space is required to complete the sort than will fit into the memory provided, then temporary segments on disk are used to hold the intermediate sort runs.

The default is adequate for most OLTP operations. You might want to adjust this parameter for decision support systems, batch jobs, or large CREATE INDEX operations.

**See Also:**

- *Oracle Database Concepts* for information on sort areas
- Your operating system-specific Oracle documentation for the default value on your system
- "SORT_AREA_RETAINED_SIZE"

## SPATIAL_VECTOR_ACCELERATION

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | Multiple instances should use the same value |

`SPATIAL_VECTOR_ACCELERATION` enables or disables the spatial vector acceleration. Setting this parameter to `true` improves spatial vector query performance.

**Note:**

This initialization parameter is only available to the Oracle Spatial and Graph option.

**See Also:**

*Oracle Spatial and Graph Developer's Guide* for more information about this parameter

## SPFILE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `SPFILE = spfile_name` |
| **Default value** | *ORACLE_HOME*`/dbs/spfile.ora` |
| **Modifiable** | No |
| **Range of values** | Any valid `SPFILE` |
| **Basic** | No |

| Property | Description |
|---|---|
| **Oracle RAC** | Multiple instances must have the same value. |

The value of this parameter is the name of the current server parameter file (SPFILE) in use. This parameter can be defined in a client side PFILE to indicate the name of the server parameter file to use.

When the default server parameter file is used by the server, the value of SPFILE is internally set by the server.

The SPFILE resides in the *ORACLE_HOME*/dbs directory; however, users can place it anywhere on their system as long as it is specified in an initialization parameter file.

**See Also:**

*Oracle Database Administrator's Guide* for more information about creating the server parameter file

# SQL_TRACE

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | true \| false |
| **Basic** | No |

SQL_TRACE enables or disables the SQL trace facility. Setting this parameter to true provides information on tuning that you can use to improve performance.

**Note:**

Using this initialization parameter to enable the SQL trace facility for the entire instance can have a severe performance impact. Enable the facility for specific sessions using the ALTER SESSION statement. If you must enable the facility on an entire production environment, then you can minimize performance impact by:

- Maintaining at least 25% idle CPU capacity
- Maintaining adequate disk space for the USER_DUMP_DEST location
- Striping disk space over sufficient disks

**Note:**

The SQL_TRACE parameter is deprecated. Oracle recommends that you use the DBMS_MONITOR and DBMS_SESSION packages instead. SQL_TRACE is retained for backward compatibility only.

**See Also:**

- "USER_DUMP_DEST"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_MONITOR package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SESSION package
- *Oracle Database Performance Tuning Guide* for more information about performance diagnostic tools

# SQL92_SECURITY

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | false |
| **Modifiable** | No |
| **Range of values** | true \| false |
| **Basic** | No |

The SQL standard specifies that security administrators should be able to require that users have SELECT privilege on a table when executing an UPDATE or DELETE statement that references table column values in a WHERE or SET clause. SQL92_SECURITY specifies whether users must have been granted the SELECT object privilege in order to execute such UPDATE or DELETE statements.

### Values

- true

  The user must have SELECT privilege on a column to reference it in the WHERE clause of a DELETE or UPDATE statement, on the right hand side of an assignment in the SET clause of an UPDATE statement.

- false

  A user with DELETE privilege on the target table of a DELETE statement may reference any column of that target table in the WHERE clause. A user with UPDATE privilege on the target table of an UPDATE statement may reference any column of that target in the WHERE clause or on the right hand side of any assignment in the SET clause of the UPDATE statement.

## SQLTUNE_CATEGORY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | SQLTUNE_CATEGORY = *category_name* |
| **Default value** | DEFAULT |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

SQLTUNE_CATEGORY specifies the category name for use by sessions to qualify the lookup of SQL profiles during SQL compilation.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for information on the DBMS_SQLTUNE package

## STANDBY_ARCHIVE_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | STANDBY_ARCHIVE_DEST = *filespec* |
| **Default value** | Operating system-specific |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | A valid path or device name other than RAW |
| **Basic** | No |

**Note:**

The STANDBY_ARCHIVE_DEST parameter is deprecated. It is retained for backward compatibility only.

Use STANDBY_ARCHIVE_DEST to specify where archived logs received from a primary database are stored on a standby database. It is no longer necessary to set this parameter, because an appropriate location is automatically chosen.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about Oracle Data Guard

# STANDBY_FILE_MANAGEMENT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `STANDBY_FILE_MANAGEMENT = { MANUAL \| AUTO }` |
| **Default value** | `MANUAL` |
| **Modifiable** | `ALTER SYSTEM` |
| **Basic** | No |

`STANDBY_FILE_MANAGEMENT` enables or disables automatic standby file management. When automatic standby file management is enabled, operating system file additions and deletions on the primary database are replicated on the standby database. `STANDBY_FILE_MANAGEMENT` is only applicable to physical standby databases.

### Values

- `MANUAL`

  disables automatic standby file management

- `AUTO`

  enables automatic standby file management

Setting `STANDBY_FILE_MANAGEMENT` to `AUTO` causes Oracle to automatically create files on the standby database and, in some cases, overwrite existing files. Care must be taken when setting `STANDBY_FILE_MANAGEMENT` and `DB_FILE_NAME_CONVERT` so that existing standby files will not be accidentally overwritten.

If the standby database is on the same system as the primary database, then ensure that the primary and standby systems do not point to the same files.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about setting this parameter

# STAR_TRANSFORMATION_ENABLED

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `STAR_TRANSFORMATION_ENABLED = { FALSE \| TRUE \| TEMP_DISABLE }` |
| **Default value** | `FALSE` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | Yes |

STAR_TRANSFORMATION_ENABLED determines whether a cost-based query transformation will be applied to star queries.

### Values

- FALSE

  The transformation will not be applied.

- TRUE

  The optimizer will consider performing a cost-based query transformation on the star query.

- TEMP_DISABLE

  The optimizer will consider performing a cost-based query transformation on the star query but will not use temporary tables in the star transformation.

**See Also:**

- *Oracle Database SQL Tuning Guide* for information on enabling star queries

## STATISTICS_LEVEL

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | STATISTICS_LEVEL = { ALL \| TYPICAL \| BASIC } |
| **Default value** | TYPICAL |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Basic** | No |

STATISTICS_LEVEL specifies the level of collection for database and operating system statistics. The Oracle Database collects these statistics for a variety of purposes, including making self-management decisions.

The default setting of TYPICAL ensures collection of all major statistics required for database self-management functionality and provides best overall performance. The default value should be adequate for most environments.

When the STATISTICS_LEVEL parameter is set to ALL, additional statistics are added to the set of statistics collected with the TYPICAL setting. The additional statistics are timed operating system statistics and plan execution statistics.

Setting the STATISTICS_LEVEL parameter to BASIC disables the collection of many of the important statistics required by Oracle Database features and functionality, including:

- Automatic Workload Repository (AWR) Snapshots
- Automatic Database Diagnostic Monitor (ADDM)
- All server-generated alerts

- Automatic SGA Memory Management
- Automatic optimizer statistics collection
- Object level statistics
- End to End Application Tracing (V$CLIENT_STATS)
- Database time distribution statistics (V$SESS_TIME_MODEL and V$SYS_TIME_MODEL)
- Service level statistics
- Buffer cache advisory
- MTTR advisory
- Shared pool sizing advisory
- Segment level statistics
- PGA Target advisory
- Timed statistics
- Monitoring of statistics

**Note:**

Oracle strongly recommends that you do not disable these important features and functionality.

When the STATISTICS_LEVEL parameter is modified by ALTER SYSTEM, all advisories or statistics are dynamically turned on or off, depending on the new value of STATISTICS_LEVEL. When modified by ALTER SESSION, the following advisories or statistics are turned on or off in the local session only. Their systemwide state is not changed:

- Timed statistics
- Timed operating system statistics
- Plan execution statistics

The V$STATISTICS_LEVEL view displays information about the status of the statistics or advisories controlled by the STATISTICS_LEVEL parameter. See "V$STATISTICS_LEVEL".

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this parameter

# STREAMS_POOL_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Big integer |
| **Syntax** | `STREAMS_POOL_SIZE = integer [K \| M \| G]` |
| **Default value** | `0` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Minimum: `0` (values greater than zero are rounded up to the nearest granule size)<br>Maximum: operating system-dependent |
| **Basic** | No |

Oracle's Automatic Shared Memory Management feature manages the size of the Streams pool when the `SGA_TARGET` initialization parameter is set to a nonzero value. If the `STREAMS_POOL_SIZE` initialization parameter also is set to a nonzero value, then Automatic Shared Memory Management uses this value as a minimum for the Streams pool.

If the `STREAMS_POOL_SIZE` initialization parameter is set to a nonzero value, and the `SGA_TARGET` parameter is set to 0 (zero), then the Streams pool size is the value specified by the `STREAMS_POOL_SIZE` parameter, in bytes.

If both the `STREAMS_POOL_SIZE` and the `SGA_TARGET` initialization parameters are set to 0 (zero), then, by default, on the first request for Streams pool memory in a database, an amount of memory equal to 10% of the shared pool is transferred from the buffer cache to the Streams pool. Products and features that use the Streams pool include Oracle GoldenGate, XStream, Oracle Streams, Oracle Advanced Queuing, and Oracle Data Pump.

The Streams pool is a shared resource, and the amount of memory a process can use from the Streams pool is determined by the application. The capture or apply parameter `MAX_SGA_SIZE` can be controlled for Oracle GoldenGate or XStream. For Oracle Advanced Queuing, use the procedures in the `dbms_aqadm` package to control the amount of Streams Pool needed.

**See Also:**

- *Oracle Database XStream Guide* for information on configuring the Streams pool for an XStream Out configuration
- *Oracle Database XStream Guide* for information on configuring the Streams pool for an XStream In configuration
- *Oracle Streams Replication Administrator's Guide*
- *Oracle GoldenGate for Windows and UNIX Administrator's Guide*
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `dbms_aqadm` package

# TAPE_ASYNCH_IO

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`TAPE_ASYNCH_IO` controls whether I/O to sequential devices (for example, backup or restore of Oracle data to or from tape) is asynchronous—that is, whether parallel server processes can overlap I/O requests with CPU processing during table scans. If your platform supports asynchronous I/O to sequential devices, Oracle recommends that you leave this parameter set to its default. However, if the asynchronous I/O implementation is not stable, you can set `TAPE_ASYNCH_IO` to `false` to disable asynchronous I/O. If your platform does not support asynchronous I/O to sequential devices, this parameter has no effect.

**See Also:**

*Oracle Database VLDB and Partitioning Guide* for more information about this parameter

# TEMP_UNDO_ENABLED

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | Each session of each instance can have its own value or not set any value at all |

`TEMP_UNDO_ENABLED` determines whether transactions within a particular session can have a temporary undo log. The default choice for database transactions has been to have a single undo log per transaction. This parameter, at the session level / system level scope, lets a transaction split its undo log into temporary undo log (for changes on temporary objects) and permanent undo log (for changes on persistent objects).

By splitting the undo stream of a transaction into two streams (temporary and permanent), a database can provide separate storage and retention model for these. This results in overall reduction in the size of undo log and redo log in the database

If database applications make use of temporary objects (using global temporary tables or temporary table transformations), it is advisable to set this parameter's value to `true`.

When `TEMP_UNDO_ENABLED` is set to `true` and the COMPATIBLE initialization parameter is set to `12.0.0`, this feature is enabled. The temporary undo feature is enabled for the session in which it is set. Setting it across the system will affect all existing and upcoming sessions. If the value is set in the `init.ora` file, all upcoming sessions will inherit this value unless overwritten by an explicit ALTER SESSION or ALTER SYSTEM statement. All undo for operations on temporary objects is deemed temporary.

If `TEMP_UNDO_ENABLED` is not set to `true`, existing applications that make use of temporary objects run as is without any change.

Once the value of the parameter is set, it cannot be changed for the lifetime of the session. If the session has temporary objects using temporary undo, the parameter cannot be disabled for the session. Similarly, if the session already has temporary objects using regular undo, setting this parameter will have no effect.

This parameter is only applicable for the primary database. For a standby database, this parameter is ignored because temporary undo is enabled by default on the standby database.

**See Also:**

*Oracle Database Administrator's Guide* for information on managing temporary undo

# THREAD

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `0` |
| **Modifiable** | No |
| **Range of values** | 0 to the maximum number of enabled threads |
| **Basic** | No |
| **Oracle RAC** | If specified, multiple instances must have different values. |

`THREAD` has been superseded by the `INSTANCE_NAME` and `INSTANCE_NUMBER` parameters, and will be made obsolete in a future release of the Oracle Database.

**See Also:**

"INSTANCE_NAME" and "INSTANCE_NUMBER"

# THREADED_EXECUTION

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | If specified, all instances must use the same value |

`THREADED_EXECUTION` specifies whether to enable the multithreaded Oracle model. Starting in Oracle Database 12*c*, the multithreaded Oracle model enables Oracle processes on UNIX and Linux to run as operating system threads in separate address spaces.

By default, some background processes on UNIX and Linux always use threaded execution; the remaining Oracle processes run as operating system processes. Thus, an "Oracle process" is not always equivalent to an "operating system process."

**Note:**

When this initialization parameter is set to `TRUE`, which enables the multithreaded Oracle model, operating system authentication is not supported. Attempts to connect to the database using operating system authentication (for example, `CONNECT / AS SYSDBA` or `CONNECT /` ) when this initialization parameter is set to `TRUE` receive an `ORA-01031 "insufficient privileges"` error.

The solution to this error is to always use the password when connecting to the database.

Also, when this initialization parameter is set to `TRUE`, the `DEDICATED_THROUGH_BROKER_listener-name=ON` parameter should be added to the listener.ora file, where *`listener-name`* is the name of the database listener. This enables the server to spawn threads when connections to the database are requested through the listener.

**See Also:**

- *Oracle Database Concepts* for more information about multithreaded Oracle
- Table F-1 for more information about background processes that run as threads instead of as operating system processes when multithreaded Oracle is enabled
- *Oracle Database Net Services Reference* for more information about the `DEDICATED_THROUGH_BROKER_`*listener-name* parameter in the listener.ora file

# TIMED_OS_STATISTICS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | If STATISTICS_LEVEL is set to ALL, then 60<br>If STATISTICS_LEVEL is set to BASIC or TYPICAL, then 0 |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | Unlimited |
| **Basic** | No |

TIMED_OS_STATISTICS specifies (in seconds) the interval at which Oracle collects operating system statistics when a request is made from the client to the server or when a request completes.

- On dedicated servers, Oracle collects operating system statistics at user logon and after each subsequent client invocation through the OCI into the Oracle server as a remote procedure call message.
- On shared servers, Oracle collects statistics when client calls to Oracle are processed.

A value of zero specifies that operating system statistics are not gathered. To collect statistics, set a value meaningful for your application and site needs.

**Note:**

Gathering operating system statistics is very expensive. Oracle recommends that you set this parameter in an ALTER SYSTEM statement rather than in the initialization parameter file, and that you reset the value to zero as soon as the needed statistics have been gathered.

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this parameter

# TIMED_STATISTICS

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | If STATISTICS_LEVEL is set to TYPICAL or ALL, then true<br>If STATISTICS_LEVEL is set to BASIC, then false |
| **Modifiable** | ALTER SESSION, ALTER SYSTEM |
| **Range of values** | true \| false |

| Property | Description |
|---|---|
| **Basic** | No |

`TIMED_STATISTICS` specifies whether statistics related to time are collected.

### Values

- `true`

  The statistics are collected and stored in trace files or displayed in the `V$SESSTATS` and `V$SYSSTATS` dynamic performance views.

- `false`

  The value of all time-related statistics is set to zero. This setting lets Oracle avoid the overhead of requesting the time from the operating system.

Starting with release 11.1.0.7.0, the value of the `TIMED_STATISTICS` parameter cannot be set to `false` if the value of `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`.

On some systems with very fast timer access, Oracle might enable timing even if this parameter is set to `false`. On these systems, setting the parameter to `true` can sometimes produce more accurate statistics for long-running operations.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information on setting this parameter
- Statistics Descriptions indicates which statistics depend on the setting of this parameter.

## TRACE_ENABLED

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `true` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | The default value is `TRUE`. Oracle recommends that multiple instances have the same value. |

`TRACE_ENABLED` controls tracing of the execution history, or code path, of Oracle. Oracle Support Services uses this information for debugging.

When `TRACE_ENABLED` is set to `true`, Oracle records information in specific files when errors occur.

Oracle records this information for all instances, even if only one instance terminates. This allows Oracle to retain diagnostics for an entire cluster.

Although the overhead incurred from this processing is not excessive, you can improve performance by setting TRACE_ENABLED to false. You might do this, for example, to meet high-end benchmark requirements. However, if you leave this parameter set to false, you may lose valuable diagnostic information. Therefore, always set TRACE_ENABLED to true to trace system problems and to reduce diagnostic efforts when unexplained instance failures occur.

# TRACEFILE_IDENTIFIER

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | TRACEFILE_IDENTIFIER = "*traceid*" |
| **Default value** | There is no default value. |
| **Modifiable** | ALTER SESSION |
| **Range of values** | Any characters that can occur as part of a file name on the customer platform |
| **Basic** | No |

TRACEFILE_IDENTIFIER specifies a custom identifier that becomes part of the Oracle Trace file name. Such a custom identifier is used to identify a trace file simply from its name and without having to open it or view its contents.

Each time this parameter is dynamically modified, the next trace dump will be written to a trace file which has the new parameter value embedded in its name. Trace file continuity information is automatically added to both the old and new trace files to indicate that these trace files belong to the same process.

This parameter can only be used to change the name of the foreground process' trace file; the background processes continue to have their trace files named in the regular format. For foreground processes, the TRACEID column of the V$PROCESS view contains the current value of the TRACEFILE_IDENTIFIER parameter. When this parameter value is set, the trace file name has the following format:

*sid*_ora_*pid*_*traceid*.trc

In this example, *sid* is the Oracle instance ID, *pid* is the process ID, and *traceid* is the value of the TRACEFILE_IDENTIFIER parameter.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about this parameter
- This parameter is not supported on all operating systems. See your operating system-specific Oracle documentation for more information.

# TRANSACTIONS

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | Derived: (1.1 * SESSIONS) |
| **Modifiable** | No |
| **Range of values** | 4 to $2^{32}$ |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

TRANSACTIONS specifies how many rollback segments to online when UNDO_MANAGEMENT = MANUAL. The maximum number of concurrent transactions is now restricted by undo tablespace size (UNDO_MANAGEMENT = AUTO) or the number of online rollback segments (UNDO_MANAGEMENT = MANUAL).

# TRANSACTIONS_PER_ROLLBACK_SEGMENT

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 5 |
| **Modifiable** | No |
| **Range of values** | 1 to operating system-dependent |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values. |

TRANSACTIONS_PER_ROLLBACK_SEGMENT specifies the number of concurrent transactions you expect each rollback segment to have to handle. The minimum number of rollback segments acquired at startup is TRANSACTIONS divided by the value for this parameter. For example, if TRANSACTIONS is 101 and this parameter is 10, then the minimum number of rollback segments acquired would be the ratio 101/10, rounded up to 11.

You can acquire more rollback segments by naming them in the parameter ROLLBACK_SEGMENTS.

**See Also:**

Your operating system-specific Oracle documentation for the range of values for this parameter.

# UNDO_MANAGEMENT

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `UNDO_MANAGEMENT = { MANUAL \| AUTO }` |
| **Default value** | `AUTO` |
| **Modifiable** | No |
| **Basic** | No |
| **Oracle RAC** | Multiple instances must have the same value. |

`UNDO_MANAGEMENT` specifies which undo space management mode the system should use. When set to `AUTO`, the instance starts in automatic undo management mode. In manual undo management mode, undo space is allocated externally as rollback segments.

**Note:**

In a CDB, the `UNDO_MANAGEMENT` initialization parameter must be set to `AUTO`, and an undo tablespace is required to be created to manage the undo data.

**See Also:**

- *Oracle Database Administrator's Guide* for more information about this parameter
- *Oracle Database Administrator's Guide* for more information about managing undo data

# UNDO_RETENTION

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | `900` |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `0` to $2^{31}$ - 1 |
| **Basic** | No |
| **Oracle RAC** | Oracle recommends that multiple instances have the same value. |

`UNDO_RETENTION` specifies (in seconds) the low threshold value of undo retention. For AUTOEXTEND undo tablespaces, the system retains undo for at least the time specified in this parameter, and automatically tunes the undo retention period to

satisfy the undo requirements of the queries. For fixed- size undo tablespaces, the system automatically tunes for the maximum possible undo retention period, based on undo tablespace size and usage history, and ignores UNDO_RETENTION unless retention guarantee is enabled.

The setting of this parameter should account for any flashback requirements of the system. Automatic tuning of undo retention is not supported for LOBs. The RETENTION value for LOB columns is set to the value of the UNDO_RETENTION parameter.

The UNDO_RETENTION parameter can only be honored if the current undo tablespace has enough space. If an active transaction requires undo space and the undo tablespace does not have available space, then the system starts reusing unexpired undo space. This action can potentially cause some queries to fail with a "snapshot too old" message.

The amount of time for which undo is retained for the Oracle Database for the current undo tablespace can be obtained by querying the TUNED_UNDORETENTION column of the V$UNDOSTAT dynamic performance view.

**See Also:**

- *Oracle Database SQL Language Reference* for information about creating undo tablespaces
- *Oracle Database Administrator's Guide* for information about managing undo data

# UNDO_TABLESPACE

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | UNDO_TABLESPACE = *undoname* |
| **Default value** | The first available undo tablespace in the database. |
| **Modifiable** | ALTER SYSTEM |
| **Range of values** | Legal name of an existing undo tablespace |
| **Basic** | Yes |
| **Oracle RAC** | Each instance must have a unique value for this parameter, when it is set. |

UNDO_TABLESPACE specifies the undo tablespace to be used when an instance starts. If this parameter is specified when the instance is in manual undo management mode, then an error will occur and startup will fail.

If the UNDO_TABLESPACE parameter is omitted, the first available undo tablespace in the database is chosen. If no undo tablespace is available, the instance will start without an undo tablespace. In such cases, user transactions will be executed using the SYSTEM rollback segment. You should avoid running in this mode under normal circumstances.

You can replace an undo tablespace with another undo tablespace while the instance is running.

**Note:**

When you update this parameter on the primary database in an Oracle Data Guard configuration, you must also update it on all the physical standby databases in the configuration. This ensures that the standby databases can find the undo tablespace when they become the primary database.

**See Also:**

*Oracle Database SQL Language Reference* for information about creating undo tablespaces

# UNIFIED_AUDIT_SGA_QUEUE_SIZE

| Property | Description |
|---|---|
| **Parameter type** | Integer |
| **Default value** | 1 MB |
| **Modifiable** | No |
| **Range of values** | 1 MB to 30 MB |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can have different values |

`UNIFIED_AUDIT_SGA_QUEUE_SIZE` specifies the size of SGA queue for unified auditing. The parameter can be useful when queued-write mode is used for unified auditing. In queued-write mode, audit records are first written to the SGA queue. When the SGA queue reaches a particular threshold, the audit records are flushed to the AUDSYS schema table. The SGA queue size should be tuned according to the audit data generation. If large numbers of audit records are generated very frequently, you can increase the size of SGA queue, so that frequent flushes of queue data can be prevented. Similarly, if fewer audit records are generated, a smaller size can be used for the SGA queue.

**See Also:**

*Oracle Database Security Guide* for more information about writing unified audit trail records to the AUDSYS schema.

# USE_DEDICATED_BROKER

| Property | Description |
|---|---|
| **Parameter type** | Boolean |

| Property | Description |
|---|---|
| **Default value** | `False` unless multithreaded Oracle is enabled[1] |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | `true \| false` |
| **Basic** | No |
| **Oracle RAC** | If specified, all instances must use the same value |

[1] Multithreaded Oracle is enabled by setting the `THREADED_EXECUTION` initialization parameter to `true`.

`USE_DEDICATED_BROKER` determines how dedicated servers are spawned. When this parameter is set to `false`, the listener spawns a dedicated server directly.

When this parameter is set to `true`, the listener hands the connection to a dedicated connection broker that spawns the dedicated server. Unlike the listener, the dedicated connection broker is a database process, and hence it can implement policies leveraging database information before the spawn.

To enable the dedicated connection broker using the `USE_DEDICATED_BROKER` initialization parameter, you must:

1. Set the `USE_DEDICATED_BROKER` initialization parameter to `true`.
2. Set the `DEDICATED_THROUGH_BROKER_`*listener-name* parameter to `on` in the listener.ora file.

Performing these two steps configures the Connection Broker Process (N*nnn*).

---

**Note:**

When multithreaded Oracle is enabled, the dedicated connection broker process is also enabled and used.

---

Configuration of the brokers is controlled by the `CONNECTION_BROKERS` initialization parameter.

---

**See Also:**

- "CONNECTION_BROKERS" for more information about configuring brokers
- Table F-1 for more information about the Connection Broker Process (N*nnn*)
- "THREADED_EXECUTION" for more information about enabling multithreaded Oracle
- *Oracle Database Net Services Reference* for more information about the `DEDICATED_THROUGH_BROKER_`*listener-name* parameter in the listener.ora file

---

# USE_INDIRECT_DATA_BUFFERS

| Property | Description |
|---|---|
| **Parameter type** | Boolean |
| **Default value** | `false` |
| **Modifiable** | No |
| **Range of values** | `true \| false` |
| **Basic** | No |

`USE_INDIRECT_DATA_BUFFERS` controls how the system global area (SGA) uses memory. It enables or disables the use of the extended buffer cache mechanism for 32-bit platforms that can support more than 4 GB of physical memory. On platforms that do not support this much physical memory, this parameter is ignored.

**See Also:**

- "LOCK_SGA", "SHARED_MEMORY_ADDRESS", and "HI_SHARED_MEMORY_ADDRESS", which are other parameters that control how the SGA uses memory.
- *Oracle Database Concepts* for more information about the SGA.

# USE_LARGE_PAGES

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `USE_LARGE_PAGES = { TRUE \| FALSE \| ONLY }` |
| **Default value** | `TRUE` |
| **Modifiable** | No |
| **Basic** | No |
| **Oracle RAC** | Multiple instances can use different values |

`USE_LARGE_PAGES` is used to manage the database's use of large pages for SGA memory. This parameter does not affect process-private memory allocations. This parameter is applicable only on the Linux operating system.

### Values

- `TRUE`

  Specifies that the instance can use large pages if large pages are configured on the system.

  In Oracle Database 11*g* Release 2 (11.2.0.2), if there are not enough large pages configured on the system, then regular sized pages will be used to allocate SGA

memory. This can cause the free large pages to go unused, and the operating system can allocate a huge amount of memory to create page tables to map SGA into physical pages for the Oracle processes. This may lead to `ORA-04030` errors and severe performance degradation on an instance.

In Oracle Database 11*g* Release 2 (11.2.0.3), Oracle allocates as much of the SGA as it can in large pages, and if it runs out, it will allocate the rest of the SGA using regular sized pages. This can cause the instance to create additional shared memory segments for the SGA, but the total SGA size will be unchanged. In this supported mixed page mode allocation, the database will exhaust the available large pages before switching to regular sized pages.

- `FALSE`

  Specifies that the instance will not use large pages. This setting is not recommended because it can cause severe performance degradation for the instance.

- `ONLY`

  Specifies that the instance will fail to start if large pages cannot be used for the entire SGA memory.

**See Also:**

- *Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems* for an overview of HugePages and information on configuring HugePages

# USER_DUMP_DEST

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `USER_DUMP_DEST = { pathname \| directory }` |
| **Default value** | Operating system-dependent |
| **Modifiable** | `ALTER SYSTEM` |
| **Range of values** | Any valid local path, directory, or disk |
| **Basic** | No |

**Note:**

The `USER_DUMP_DEST` initialization parameter is deprecated.

`USER_DUMP_DEST` specifies the pathname for a directory where the server will write debugging trace files on behalf of a user process.

For example, this directory might be set as follows:

- On MS-DOS: `C:\ORACLE\UTRC`

- On UNIX: `/oracle/utrc`
- On VMS: `DISK$UR3:[ORACLE.UTRC]`

**Note:**

This parameter is ignored by the diagnosability infrastructure introduced in Oracle Database 11g Release 1 (11.1), which places trace and core files in a location controlled by the `DIAGNOSTIC_DEST` initialization parameter.

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about the use of trace files
- Your operating system-specific Oracle documentation for the range of values

# UTL_FILE_DIR

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `UTL_FILE_DIR = pathname` |
| **Default value** | There is no default value. |
| **Modifiable** | No |
| **Range of values** | Any valid directory path |
| **Basic** | No |

`UTL_FILE_DIR` lets you specify one or more directories that Oracle should use for PL/SQL file I/O. If you are specifying multiple directories, you must repeat the `UTL_FILE_DIR` parameter for each directory on separate lines of the initialization parameter file.

All users can read or write to all files specified by this parameter. Therefore all PL/SQL users must be trusted with the information in the directories specified by this parameter.

**Note:**

If you list multiple values, all entries of this parameter must be on contiguous lines of the parameter file. If you separate them with other parameters, Oracle will read only the last (contiguous) lines.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for information about the recommended alternative to using the `UTL_FILE_DIR` parameter

## WORKAREA_SIZE_POLICY

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `WORKAREA_SIZE_POLICY = { AUTO \| MANUAL }` |
| **Default value** | `AUTO` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |

`WORKAREA_SIZE_POLICY` specifies the policy for sizing work areas. This parameter controls the mode in which working areas are tuned.

### Values

- `AUTO`

  Work areas used by memory-intensive operators are sized automatically, based on the PGA memory used by the system, the target PGA memory set in `PGA_AGGREGATE_TARGET`, and the requirement of each individual operator.

- `MANUAL`

  The sizing of work areas is manual and based on the values of the `*_AREA_SIZE` parameter corresponding to the operation (for example, a sort uses `SORT_AREA_SIZE`). Specifying `MANUAL` may result in sub-optimal performance and poor PGA memory utilization.

**See Also:**

*Oracle Database SQL Tuning Guide* for information on setting this parameter when a query that calls for a hash join experiences excessive I/O

## XML_DB_EVENTS

| Property | Description |
|---|---|
| **Parameter type** | String |
| **Syntax** | `XML_DB_EVENTS = { enable \| disable }` |
| **Default value** | `enable` |
| **Modifiable** | `ALTER SESSION`, `ALTER SYSTEM` |
| **Basic** | No |

XML_DB_EVENTS enables or disables XML DB Events.

Changing this parameter through an ALTER SESSION statement affects only the current session. Only users with the XDBADMIN role are allowed to change this parameter in a session.

Changing this parameter through an ALTER SYSTEM statement will make a systemwide change of this parameter; however, the change is only registered by new sessions started after the change. Currently running sessions need to be restarted to pick up the new change. Users must have the ALTER SYSTEM privilege to make a systemwide change.

**See Also:**

*Oracle XML DB Developer's Guide* for more information about this parameter

# Part II

# Static Data Dictionary Views

This part describes data dictionary tables and views. These tables and views are called **static**, because they change only when a change is made to the data dictionary (for example, when a new table is created or when a user is granted new privileges). This part contains the following chapters:

- Static Data Dictionary Views: ALL_ALL_TABLES to ALL_OUTLINES
- Static Data Dictionary Views: ALL_PART_COL_STATISTICS to DATABASE_PROPERTIES
- Static Data Dictionary Views: DBA_2PC_NEIGHBORS to DBA_HIST_JAVA_POOL_ADVICE
- Static Data Dictionary Views: DBA_HIST_LATCH to DBA_STORED_SETTINGS
- Static Data Dictionary Views: DBA_STREAMS_ADD_COLUMN to USER_ZONEMAPS

---

**Note:**

Oracle also maintains tables that monitor ongoing database activity. These **dynamic performance tables** are described in Dynamic Performance Views .

---

# 2

# Static Data Dictionary Views: ALL_ALL_TABLES to ALL_OUTLINES

This chapter describes the first set (in alphabetical order) of static data dictionary views.The remaining static data dictionary views appear in alphabetical order in Static Data Dictionary Views: ALL_PART_COL_STATISTICS to DATABASE_PROPERTIES through Static Data Dictionary Views: DBA_STREAMS_ADD_COLUMN to USER_ZONEMAPS.

This chapter contains the following topics:


- About Static Data Dictionary Views
- CDB_* Views
- Oracle Database Real Application Security Views
- Oracle Label Security Views
- Oracle Database Vault Views
- Oracle Replication Views
- Oracle Workspace Manager Views
- Recovery Catalog Views
- Static Data Dictionary View Descriptions


## About Static Data Dictionary Views

Data dictionary tables are not directly accessible, but you can access information in them through data dictionary views. To list the data dictionary views available to you, query the view `DICTIONARY`.

Many data dictionary tables have three corresponding views:

- An `ALL_` view displays all the information accessible to the current user, including information from the current user's schema as well as information from objects in other schemas, if the current user has access to those objects by way of grants of privileges or roles.
- A `DBA_` view displays all relevant information in the entire database. `DBA_` views are intended only for administrators. They can be accessed only by users with the `SELECT ANY DICTIONARY` privilege. This privilege is assigned to the `DBA` role when the system is initially installed.
- A `USER_` view displays all the information from the schema of the current user. No special privileges are required to query these views.

The columns of the ALL_, DBA_, and USER_ views corresponding to a single data dictionary table are usually nearly identical. Therefore, these views are described in full only once in this chapter, at their first occurrence alphabetically, and are listed without full descriptions at their other occurrences.

**See Also:**

"Static Data Dictionary View Descriptions" introduces the alphabetical listing of view descriptions that are documented in this manual

## CDB_* Views

For every DBA_* view, a CDB_* view is defined. In the root of a multitenant container database (CDB), CDB_* views can be used to obtain information about tables, tablespaces, users, privileges, parameters, and so on contained in the root and in pluggable databases (PDBs).

CDB_* views are container data objects. When a user connected to the root queries a CDB_* view, the query results will depend on the CONTAINER_DATA attribute for users for the view. The CONTAINER_DATA clause of the SQL ALTER USER statement is used to set and modify users' CONTAINER_DATA attribute.

The CDB_* views are owned by SYS, regardless of who owns the underlying DBA_* view.

By default, a user connected to the root will only see data pertaining to the root.

**See Also:**

- *Oracle Database Security Guide* for more information about container data objects
- *Oracle Database SQL Language Reference* for more information about the CONTAINER_DATA clause for the SQL ALTER USER statement

In a PDB, the CDB_* views only show objects visible through a corresponding DBA_* view.

In addition to all the columns found in a given DBA_* view, the corresponding CDB_* view also contains the CON_ID column, which identifies a container whose data a given CDB_* row represents. In a non-CDB, the value of a CON_ID column will be 0.

Data returned by these views depends on whether a given PDB is open at the time the query is issued. In particular, in an Oracle RAC environment, data returned by these view may vary according to the instance to which a session is connected.

## Oracle Database Real Application Security Views

This manual describes the following Oracle Database Real Application Security auditing views:

DBA_XS_AUDIT_POLICY_OPTIONS
DBA_XS_AUDIT_TRAIL
DBA_XS_ENB_AUDIT_POLICIES

Descriptions of the other Oracle Database Real Application Security views are provided in *Oracle Database Real Application Security Administrator's and Developer's Guide*.

**Note:**

Oracle Database Real Application Security views include _XS_ in the view name.

## Oracle Label Security Views

Descriptions of Oracle Label Security views are provided in *Oracle Label Security Administrator's Guide*.

**Note:**

Oracle Label Security views include _SA_ in the view name.

## Oracle Database Vault Views

Descriptions of Oracle Database Vault views are provided in *Oracle Database Vault Administrator's Guide*.

**Note:**

Oracle Database Vault views include _DV_ in the view name.

## Oracle Replication Views

A number of data dictionary views are relevant only if you are using Oracle Replication. Table 2-1 lists these views.

***Table 2-1 Oracle Replication Data Dictionary Views***

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| ALL_REPCAT | DBA_REPCAT | USER_REPCAT |
| | DBA_REPCAT_EXCEPTIONS | |
| ALL_REPCAT_TEMPLATE_OBJECTS | DBA_REPCAT_TEMPLATE_OBJECTS | USER_REPCAT_TEMPLATE_OBJECTS |
| ALL_REPCAT_TEMPLATE_PARMS | DBA_REPCAT_TEMPLATE_PARMS | USER_REPCAT_TEMPLATE_PARMS |
| ALL_REPCAT_TEMPLATE_SITES | DBA_REPCAT_TEMPLATE_SITES | USER_REPCAT_TEMPLATE_SITES |
| ALL_REPCAT_USER_AUTHORIZATIONS | DBA_REPCAT_USER_AUTHORIZATIONS | USER_REPCAT_USER_AUTHORIZATIONS |
| ALL_REPCAT_USER_PARM_VALUES | DBA_REPCAT_USER_PARM_VALUES | USER_REPCAT_USER_PARM_VALUES |
| ALL_REPCATLOG | DBA_REPCATLOG | USER_REPCATLOG |

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| ALL_REPCOLUMN | DBA_REPCOLUMN | USER_REPCOLUMN |
| ALL_REPCOLUMN_GROUP | DBA_REPCOLUMN_GROUP | USER_REPCOLUMN_GROUP |
| ALL_REPCONFLICT | DBA_REPCONFLICT | USER_REPCONFLICT |
| ALL_REPDDL | DBA_REPDDL | USER_REPDDL |
| | DBA_REPEXTENSIONS | |
| ALL_REPFLAVOR_COLUMNS | DBA_REPFLAVOR_COLUMNS | USER_REPFLAVOR_COLUMNS |
| ALL_REPFLAVOR_OBJECTS | DBA_REPFLAVOR_OBJECTS | USER_REPFLAVOR_OBJECTS |
| ALL_REPFLAVORS | DBA_REPFLAVORS | USER_REPFLAVORS |
| ALL_REPGENERATED | DBA_REPGENERATED | USER_REPGENERATED |
| ALL_REPGENOBJECTS | DBA_REPGENOBJECTS | USER_REPGENOBJECTS |
| ALL_REPGROUP | DBA_REPGROUP | USER_REPGROUP |
| ALL_REPGROUP_PRIVILEGES | DBA_REPGROUP_PRIVILEGES | USER_REPGROUP_PRIVILEGES |
| ALL_REPGROUPED_COLUMN | DBA_REPGROUPED_COLUMN | USER_REPGROUPED_COLUMN |
| ALL_REPKEY_COLUMNS | DBA_REPKEY_COLUMNS | USER_REPKEY_COLUMNS |
| ALL_REPOBJECT | DBA_REPOBJECT | USER_REPOBJECT |
| ALL_REPPARAMETER_COLUMN | DBA_REPPARAMETER_COLUMN | USER_REPPARAMETER_COLUMN |
| ALL_REPPRIORITY | DBA_REPPRIORITY | USER_REPPRIORITY |
| ALL_REPPRIORITY_GROUP | DBA_REPPRIORITY_GROUP | USER_REPPRIORITY_GROUP |
| ALL_REPPROP | DBA_REPPROP | USER_REPPROP |
| ALL_REPRESOL_STATS_CONTROL | DBA_REPRESOL_STATS_CONTROL | USER_REPRESOL_STATS_CONTROL |
| ALL_REPRESOLUTION | DBA_REPRESOLUTION | USER_REPRESOLUTION |
| ALL_REPRESOLUTION_METHOD | DBA_REPRESOLUTION_METHOD | USER_REPRESOLUTION_METHOD |
| ALL_REPRESOLUTION_STATISTICS | DBA_REPRESOLUTION_STATISTICS | USER_REPRESOLUTION_STATISTICS |
| ALL_REPSCHEMA | DBA_REPSCHEMA | USER_REPSCHEMA |
| ALL_REPSITES | DBA_REPSITES | USER_REPSITES |
| | DBA_REPSITES_NEW | |

The following are additional Oracle Replication data dictionary views:

DEFCALL
DEFCALLDEST
DEFDEFAULTDEST
DEFERRCOUNT
DEFERROR

DEFLOB
DEFPROPAGATOR
DEFSCHEDULE
DEFTRAN
DEFTRANDEST

**See Also:**

*Oracle Database Advanced Replication Management API Reference* for information about these views

## Oracle Workspace Manager Views

A number of data dictionary views are relevant only if you are using Oracle Workspace Manager. Table 2-2 lists these views.

***Table 2-2 Oracle Workspace Manager Data Dictionary Views***

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| ALL_MP_GRAPH_WORKSPACES | | USER_MP_GRAPH_WORKSPACES |
| ALL_MP_PARENT_WORKSPACES | | USER_MP_PARENT_WORKSPACES |
| ALL_REMOVED_WORKSPACES | | USER_REMOVED_WORKSPACES |
| ALL_VERSION_HVIEW | | |
| ALL_WM_CONS_COLUMNS | | USER_WM_CONS_COLUMNS |
| ALL_WM_CONSTRAINTS | | USER_WM_CONSTRAINTS |
| ALL_WM_IND_COLUMNS | | USER_WM_IND_COLUMNS |
| ALL_WM_IND_EXPRESSIONS | | USER_WM_IND_EXPRESSIONS |
| ALL_WM_LOCKED_TABLES | | USER_WM_LOCKED_TABLES |
| ALL_WM_MODIFIED_TABLES | | USER_WM_MODIFIED_TABLES |
| | | USER_WM_PRIVS |
| ALL_WM_RIC_INFO | | USER_WM_RIC_INFO |
| | DBA_WM_SYS_PRIVS | |
| ALL_WM_TAB_TRIGGERS | | USER_WM_TAB_TRIGGERS |
| ALL_WM_VERSIONED_TABLES | | USER_WM_VERSIONED_TABLES |
| ALL_WM_VT_ERRORS | DBA_WM_VT_ERRORS | USER_WM_VT_ERRORS |
| ALL_WORKSPACE_PRIVS | | USER_WORKSPACE_PRIVS |
| ALL_WORKSPACE_SAVEPOINTS | | USER_WORKSPACE_SAVEPOINTS |
| | DBA_WORKSPACE_SESSIONS | |

| ALL_ Views | DBA_ Views | USER_ Views |
|---|---|---|
| ALL_WORKSPACES | DBA_WORKSPACES | USER_WORKSPACES |

The following are additional Oracle Workspace Manager data dictionary views:

ROLE_WM_PRIVS
WM_COMPRESS_BATCH_SIZES
WM_COMPRESSIBLE_TABLES
WM_EVENTS_INFO
WM_INSTALLATION
WM_REPLICATION_INFO

**See Also:**

*Oracle Database Workspace Manager Developer's Guide* for information about these views

## Recovery Catalog Views

The following data dictionary views are only available after you create an optional recovery catalog (which contains schemas containing information about backups) for use with Recovery Manager:

RC_ARCHIVED_LOG
RC_BACKUP_ARCHIVELOG_DETAILS
RC_BACKUP_ARCHIVELOG_SUMMARY
RC_BACKUP_CONTROLFILE
RC_BACKUP_CONTROLFILE_DETAILS
RC_BACKUP_CONTROLFILE_SUMMARY
RC_BACKUP_COPY_DETAILS
RC_BACKUP_COPY_SUMMARY
RC_BACKUP_CORRUPTION
RC_BACKUP_DATAFILE
RC_BACKUP_DATAFILE_DETAILS
RC_BACKUP_DATAFILE_SUMMARY
RC_BACKUP_FILES
RC_BACKUP_PIECE
RC_BACKUP_PIECE_DETAILS
RC_BACKUP_REDOLOG
RC_BACKUP_SET
RC_BACKUP_SET_DETAILS
RC_BACKUP_SET_SUMMARY
RC_BACKUP_SPFILE
RC_BACKUP_SPFILE_DETAILS
RC_BACKUP_SPFILE_SUMMARY
RC_CHECKPOINT
RC_CONTROLFILE_COPY
RC_COPY_CORRUPTION
RC_DATABASE

```
RC_DATABASE_BLOCK_CORRUPTION
RC_DATABASE_INCARNATION
RC_DATAFILE
RC_DATAFILE_COPY
RC_LOG_HISTORY
RC_OFFLINE_RANGE
RC_PROXY_ARCHIVEDLOG
RC_PROXY_ARCHIVELOG_DETAILS
RC_PROXY_ARCHIVELOG_SUMMARY
RC_PROXY_CONTROLFILE
RC_PROXY_COPY_DETAILS
RC_PROXY_COPY_SUMMARY
RC_PROXY_DATAFILE
RC_REDO_LOG
RC_REDO_THREAD
RC_RESTORE_POINT
RC_RESYNC
RC_RMAN_BACKUP_JOB_DETAILS
RC_RMAN_BACKUP_SUBJOB_DETAILS
RC_RMAN_BACKUP_TYPE
RC_RMAN_CONFIGURATION
RC_RMAN_OUTPUT
RC_RMAN_STATUS
RC_SITE
RC_STORED_SCRIPT
RC_STORED_SCRIPT_LINE
RC_TABLESPACE
RC_TEMPFILE
RC_UNUSABLE_BACKUPFILE_DETAILS
```

**See Also:**

*Oracle Database Backup and Recovery Reference* for information about these views

## Static Data Dictionary View Descriptions

The remainder of this chapter describes the static data dictionary views in alphabetical order.

# ALL_ALL_TABLES

ALL_ALL_TABLES describes the object tables and relational tables accessible to the current user.

### Related Views

- DBA_ALL_TABLES describes all object tables and relational tables in the database.

- USER_ALL_TABLES describes the object tables and relational tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the table; NULL for partitioned, temporary, and index-organized tables |
| CLUSTER_NAME | VARCHAR2(128) | | Name of the cluster, if any, to which the table belongs |
| IOT_NAME | VARCHAR2(128) | | Name of the index-organized table, if any, to which the overflow or mapping table entry belongs. If the IOT_TYPE column is not NULL, then this column contains the base table name. |
| STATUS | VARCHAR2(8) | | If a previous DROP TABLE operation failed, indicates whether the table is unusable (UNUSABLE) or valid (VALID) |
| PCT_FREE | NUMBER | | Minimum percentage of free space in a block; NULL for partitioned tables |
| PCT_USED | NUMBER | | Minimum percentage of used space in a block; NULL for partitioned tables |
| INI_TRANS | NUMBER | | Initial number of transactions; NULL for partitioned tables |
| MAX_TRANS | NUMBER | | Maximum number of transactions; NULL for partitioned tables |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent (in bytes); NULL for partitioned tables |
| NEXT_EXTENT | NUMBER | | Size of secondary extents (in bytes); NULL for partitioned tables |
| MIN_EXTENTS | NUMBER | | Minimum number of extents allowed in the segment; NULL for partitioned tables |
| MAX_EXTENTS | NUMBER | | Maximum number of extents allowed in the segment; NULL for partitioned tables |
| PCT_INCREASE | NUMBER | | Percentage increase in extent size; NULL for partitioned tables |
| FREELISTS | NUMBER | | Number of process freelists allocated to the segment; NULL for partitioned tables |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated to the segment |
| LOGGING | VARCHAR2(3) | | Indicates whether or not changes to the table are logged:<br>• YES<br>• NO |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BACKED_UP | VARCHAR2(1) | | Indicates whether the table has been backed up since the last modification (Y) or not (N) |
| NUM_ROWS | NUMBER | | Number of rows in the table |
| BLOCKS | NUMBER | | Number of used blocks in the table |
| EMPTY_BLOCKS | NUMBER | | Number of empty (never used) blocks in the table |
| AVG_SPACE | NUMBER | | Average available free space in the table |
| CHAIN_CNT | NUMBER | | Number of rows in the table that are chained from one data block to another or that have migrated to a new block, requiring a link to preserve the old rowid. This column is updated only after you analyze the table. |
| AVG_ROW_LEN | NUMBER | | Average row length, including row overhead |
| AVG_SPACE_FREELIST_BLOCKS | NUMBER | | Average freespace of all blocks on a freelist |
| NUM_FREELIST_BLOCKS | NUMBER | | Number of blocks on the freelist |
| DEGREE | VARCHAR2(10) | | Number of threads per instance for scanning the table, or DEFAULT |
| INSTANCES | VARCHAR2(10) | | Number of instances across which the table is to be scanned, or DEFAULT |
| CACHE | VARCHAR2(5) | | Indicates whether the table is to be cached in the buffer cache (Y) or not (N) |
| TABLE_LOCK | VARCHAR2(8) | | Indicates whether table locking is enabled (ENABLED) or disabled (DISABLED) |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing the table |
| LAST_ANALYZED | DATE | | Date on which the table was most recently analyzed |
| PARTITIONED | VARCHAR2(3) | | Indicates whether the table is partitioned (YES) or not (NO) |
| IOT_TYPE | VARCHAR2(12) | | If the table is an index-organized table, then IOT_TYPE is IOT, IOT_OVERFLOW, or IOT_MAPPING. If the table is not an index-organized table, then IOT_TYPE is NULL. |
| OBJECT_ID_TYPE | VARCHAR2(16) | | Indicates whether the object ID (OID) is USER-DEFINED or SYSTEM GENERATED |
| TABLE_TYPE_OWNER | VARCHAR2(128) | | If an object table, owner of the type from which the table is created |
| TABLE_TYPE | VARCHAR2(128) | | If an object table, type of the table |
| TEMPORARY | VARCHAR2(1) | | Indicates whether the table is temporary (Y) or not (N) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SECONDARY | VARCHAR2(1) | | Indicates whether the table is a secondary object created by the `ODCIIndexCreate` method of the Oracle Data Cartridge to contain the contents of a domain index (`Y`) or not (`N`) |
| NESTED | VARCHAR2(3) | | Indicates whether the table is a nested table (`YES`) or not (`NO`) |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool to be used for table blocks:<br>• `DEFAULT`<br>• `KEEP`<br>• `RECYCLE`<br>• `NULL` |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for table blocks:<br>• `DEFAULT`<br>• `KEEP`<br>• `NONE`<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for table blocks:<br>• `DEFAULT`<br>• `KEEP`<br>• `NONE`<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| ROW_MOVEMENT | VARCHAR2(8) | | If a partitioned table, indicates whether row movement is enabled (`ENABLED`) or disabled (`DISABLED`) |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (`YES`) or they were not gathered (`NO`). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to `NO`. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (`YES`) or not (`NO`) |
| DURATION | VARCHAR2(15) | | Indicates the duration of a temporary table:<br>`SYS$SESSION` - Rows are preserved for the duration of the session<br>`SYS$TRANSACTION` - Rows are deleted after `COMMIT`<br>Null - Permanent table |
| SKIP_CORRUPT | VARCHAR2(8) | | Indicates whether Oracle Database ignores blocks marked corrupt during table and index scans (`ENABLED`) or raises an error (`DISABLED`). To enable this feature, run the `DBMS_REPAIR.skip_corrupt_blocks` procedure. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MONITORING | VARCHAR2(3) | | Indicates whether the table has the MONITORING attribute set (YES) or not (NO) |
| CLUSTER_OWNER | VARCHAR2(128) | | Owner of the cluster, if any, to which the table belongs |
| DEPENDENCIES | VARCHAR2(8) | | Indicates whether row-level dependency tracking is enabled (ENABLED) or disabled (DISABLED) |
| COMPRESSION | VARCHAR2(8) | | Indicates whether table compression is enabled (ENABLED) or not (DISABLED); NULL for partitioned tables |
| COMPRESS_FOR | VARCHAR2(30) | | Default compression for what kind of operations:<br>• BASIC<br>• ADVANCED<br>• QUERY LOW<br>• QUERY HIGH[1]<br>• ARCHIVE LOW[1]<br>• ARCHIVE HIGH[1]<br>• NULL |
| DROPPED | VARCHAR2(3) | | Indicates whether the table has been dropped and is in the recycle bin (YES) or not (NO); NULL for partitioned tables |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the table segment has been created (YES) or not (NO) |
| INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is enabled (ENABLED) or disabled (DISABLED) for this table |
| INMEMORY_PRIORITY[2] | VARCHAR2(8) | | Indicates the priority at which this table is populated into the In-Memory Column Store (IM column store). Possible values:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| INMEMORY_DISTRIBUTE[2] | VARCHAR2(15) | | Indicates how the table will be distributed in the IM column stores in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| INMEMORY_COMPRESSION[2] | VARCHAR2(17) | | Compression level for the in-memory store:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ] |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NULL[3] |
| INMEMORY_DUPLICATE[2] | VARCHAR2(13) | | Indicates the duplicate setting for the In-Memory Column Store (IM column store) in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |

1 Hybrid Columnar Compression is a feature of the Enterprise Edition of Oracle Database that is dependent on the underlying storage system. See *Oracle Database Concepts* for more information.

2 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

3 This column has a value based on where the segments lie for a table. For example, if the table is partitioned and is enabled for the IM column store, the value is NULL for ALL_TABLES but non-NULL for ALL_TAB_PARTITIONS.

**See Also:**

- "DBA_ALL_TABLES"
- "USER_ALL_TABLES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_REPAIR.SKIP_CORRUPT_BLOCKS procedure
- *Oracle Database Concepts* for overview information on the IM column store
- *Oracle Database Administrator's Guide* for information on using the IM column store

# ALL_APPLY

ALL_APPLY displays information about the apply processes that dequeue messages from queues accessible to the current user.

### Related View

DBA_APPLY displays information about all apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| APPLY_NAME | VARCHAR2(128) | NOT NULL | Name of the apply process |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue from which the apply process dequeues |
| QUEUE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue from which the apply process dequeues |
| APPLY_CAPTURED | VARCHAR2(3) | | Indicates whether the apply process applies captured messages (YES) or user-enqueued messages (NO) |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the positive rule set used by the apply process for filtering |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the positive rule set used by the apply process for filtering |
| APPLY_USER | VARCHAR2(128) | | User who is applying messages |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | Database link to which changes are applied. If NULL, then changes are applied to the local database. |
| APPLY_TAG | RAW(2000) | | Tag associated with redo log records that are generated when changes are made by the apply process |
| DDL_HANDLER | VARCHAR2(98) | | Name of the user-specified data definition language (DDL) handler, which handles DDL logical change records (LCRs) |
| PRECOMMIT_HANDLER | VARCHAR2(98) | | Name of the user-specified pre-commit handler |
| MESSAGE_HANDLER | VARCHAR2(98) | | Name of the user-specified procedure that handles dequeued messages other than logical change records (LCRs) |
| STATUS | VARCHAR2(8) | | Status of the apply process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| MAX_APPLIED_MESSAGE_NUMBER | NUMBER | | System change number (SCN) corresponding to the apply process high watermark for the last time the apply process was stopped using the DBMS_APPLY_ADM.STOP_APPLY procedure with the force parameter set to false. The apply process high watermark is the SCN beyond which no messages have been applied. |
| NEGATIVE_RULE_SET_NAME | VARCHAR2(128) | | Name of the negative rule set used by the apply process for filtering |
| NEGATIVE_RULE_SET_OWNER | VARCHAR2(128) | | Owner of the negative rule set used by the apply process for filtering |
| STATUS_CHANGE_TIME | DATE | | Time that the STATUS of the apply process was changed |
| ERROR_NUMBER | NUMBER | | Error number if the apply process was aborted |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message if the apply process was aborted |
| MESSAGE_DELIVERY_MODE | VARCHAR2(10) | | Reserved for internal use |
| PURPOSE | VARCHAR2(19) | | Purpose of the apply process:<br>• GoldenGate Apply - An Oracle GoldenGate Inbound server configured by Oracle GoldenGate integrated replicat<br>• Streams - An apply process in an Oracle Streams configuration |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • XStream Streams - An apply process in an Oracle Streams configuration with XStream capabilities enabled by the DBMS_XSTREAM_ADM.ENABLE_GG_XSTREAM_FOR_STREAMS procedure<br>• XStream Out - An XStream outbound server in an XStream Out configuration<br>• XStream In - An XStream inbound server in an XStream In configuration<br>• AUDIT VAULT - An apply process in an audit vault configuration<br>• CHANGE DATA CAPTURE - An apply process in a change data capture configuration |

**See Also:**

- "DBA_APPLY"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLY_ADM.STOP_APPLY procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XSTREAM_ADM.ENABLE_GG_XSTREAM_FOR_STREAMS procedure

# ALL_APPLY_CHANGE_HANDLERS

ALL_APPLY_CHANGE_HANDLERS displays information about the change handlers on the tables accessible to the current user.

### Related View

DBA_APPLY_CHANGE_HANDLERS displays information about the change handlers on all tables in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CHANGE_TABLE_OWNER | VARCHAR2(128) | | Owner of the change table |
| CHANGE_TABLE_NAME | VARCHAR2(128) | | Name of the change table |
| SOURCE_TABLE_OWNER | VARCHAR2(128) | | Owner of the source table |
| SOURCE_TABLE_NAME | VARCHAR2(128) | | Name of the source table |
| HANDLER_NAME | VARCHAR2(128) | | Name of the statement-based change handler |
| CAPTURE_VALUES | VARCHAR2(3) | | Indicates whether to capture the old (OLD), new (NEW), or both (*) values |
| APPLY_NAME | VARCHAR2(128) | | Name of the apply process |
| OPERATION_NAME | VARCHAR2(10) | | Name of the DML operation to which the DML handler is set:<br>• DEFAULT |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • INSERT<br>• UPDATE<br>• DELETE<br>• LOB_UPDATE |
| CREATION_TIME | TIMESTAMP(6) | | Change handler creation time |
| MODIFICATION_TIME | TIMESTAMP(6) | | Change handler modification time |

**See Also:**

- "DBA_APPLY_CHANGE_HANDLERS"
- *Oracle Streams Concepts and Administration* for more information about change handlers used by an Oracle Streams apply process

# ALL_APPLY_CONFLICT_COLUMNS

ALL_APPLY_CONFLICT_COLUMNS displays information about the conflict handlers on the tables accessible to the current user.

### Related View

DBA_APPLY_CONFLICT_COLUMNS displays information about the conflict handlers on all tables in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the object on which the update conflict handler is defined |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object on which the update conflict handler is defined |
| METHOD_NAME | VARCHAR2(92) | | Name of the update conflict handler used to resolve conflicts |
| RESOLUTION_COLUMN | VARCHAR2(4000) | | Name of the column used to resolve conflicts |
| COLUMN_NAME | VARCHAR2(128) | | Name of a column in the column list for the update conflict handler |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | Database link to which changes are applied. If null, then changes are applied to the local database. |

**See Also:**

- "DBA_APPLY_CONFLICT_COLUMNS"
- *Oracle Streams Concepts and Administration* for more information about Oracle Streams conflict handlers

# ALL_APPLY_DML_HANDLERS

ALL_APPLY_DML_HANDLERS displays information about the DML handlers on the tables accessible to the current user.

### Related View

DBA_APPLY_DML_HANDLERS displays information about the DML handlers on all tables in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object on which the DML handler is specified |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object on which the DML handler is specified |
| OPERATION_NAME | VARCHAR2(13) | | Name of the DML operation for which the DML handler is used:<br>• DEFAULT<br>• INSERT<br>• UPDATE<br>• DELETE<br>• LOB_UPDATE<br>• ASSEMBLE_LOBS |
| USER_PROCEDURE | VARCHAR2(98) | | Name of the user-specified DML handler, which handles row logical change records that contain the DML operation in the OPERATION_NAME column on the object |
| ERROR_HANDLER | VARCHAR2(1) | | Indicates whether the DML handler handles only the relevant row logical change records that result in apply errors (Y) or all relevant row logical change records (N) |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | Database link to which changes are applied. If null, then changes are applied to the local database. |
| APPLY_NAME | VARCHAR2(128) | | Name of the apply process for the given object |
| ASSEMBLE_LOBS | VARCHAR2(1) | | Indicates whether LOB assembly is used for LOB columns in logical change records (LCRs) processed by the handler (Y) or not (N)<br>LOB assembly combines multiple LCRs for a LOB column resulting from a single row change into one row LCR before passing the LCR to the handler. |
| SET_BY | VARCHAR2(10) | | Entity that set up the handler. Possible values include:<br>• GOLDENGATE<br>• USER |

**See Also:**

- "DBA_APPLY_DML_HANDLERS"
- *Oracle Streams Concepts and Administration* for more information about DML handlers in an Oracle Streams environment
- *Oracle Database XStream Guide* for more information about DML handlers in an Oracle XStream environment

# ALL_APPLY_ENQUEUE

ALL_APPLY_ENQUEUE displays information about the apply enqueue actions for the rules where the destination queue exists and is accessible to the current user.

### Related View

DBA_APPLY_ENQUEUE displays information about the apply enqueue actions for all rules in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| DESTINATION_QUEUE_NAME | VARCHAR2(4000) | | Name of the queue where events satisfying the rule will be enqueued |

**See Also:**

- "DBA_APPLY_ENQUEUE"
- *Oracle Streams Concepts and Administration* for instructions on specifying that an apply process enqueue messages
- *Oracle Streams Concepts and Administration* for information about enqueue destinations for messages during apply

# ALL_APPLY_ERROR

ALL_APPLY_ERROR displays information about the error transactions generated by the apply processes that dequeue messages from queues accessible to the current user.

### Related Views

- DBA_APPLY_ERROR displays information about the error transactions generated by all apply processes in the database.
- USER_APPLY_ERROR displays information about the error transactions generated by apply processes visible to the current user. This view does not display the SOURCE_ROOT_NAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| APPLY_NAME | VARCHAR2(128) | | Name of the apply process at the local database which processed the transaction |
| QUEUE_NAME | VARCHAR2(128) | | Name of the queue at the local database from which the transaction was dequeued |
| QUEUE_OWNER | VARCHAR2(128) | | Owner of the queue at the local database from which the transaction was dequeued |
| LOCAL_TRANSACTION_ID | VARCHAR2(22) | | Local transaction ID for the error transaction |
| SOURCE_DATABASE | VARCHAR2(128) | | Database where the transaction originated |
| SOURCE_TRANSACTION_ID | VARCHAR2(128) | | Original transaction ID at the source database |
| SOURCE_COMMIT_SCN | NUMBER | | Original commit system change number (SCN) for the transaction at the source database |
| MESSAGE_NUMBER | NUMBER | | Identifier for the message in the transaction that raised an error |
| ERROR_NUMBER | NUMBER | | Error number of the error raised by the transaction |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message of the error raised by the transaction |
| RECIPIENT_ID | NUMBER | | User ID of the original user that applied the transaction |
| RECIPIENT_NAME | VARCHAR2(128) | | Name of the original user that applied the transaction |
| MESSAGE_COUNT | NUMBER | | Total number of messages inside the error transaction |
| ERROR_CREATION_TIME | DATE | | Time that the error was created |
| SOURCE_COMMIT_POSITION | RAW(64) | | Original commit position for the transaction |
| ERROR_TYPE | VARCHAR2(11) | | NULL if the apply process can access all of the LCRs in the error transaction. When the ERROR_TYPE is NULL, manage the error transactions using the instructions in *Oracle Streams Concepts and Administration*.<br>EAGER ERROR if the apply process cannot access all of the LCRs in the error transaction. This error type typically means that the apply process was applying LCRs in a large transaction. When the ERROR_TYPE is EAGER ERROR, manage the error transaction using the instructions in *Oracle Database XStream Guide*.<br>RECORD LCR indicates that a single LCR has been recorded as requested by user-specified error handling configuration |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | `RECORD TXN NO LCRS` indicates that the identified transaction encountered an error and only the transaction ID is recorded as requested by user-specified error handling configuration<br>`RECORD TXN WITH LCRS` indicates that the identified transaction encountered and error. The entire transaction is recorded as requested by user-specified error handling configuration.<br>`UNHANDLED ERRORS NO LCR` indicates that the identified transaction encountered an error and there was no error handling specified for this handler. No LCRs are recorded for this transaction. |
| `SOURCE_ROOT_NAME` | `VARCHAR2(128)` | | The global name of the source root database |
| `ERROR_POSITION` | `RAW(64)` | | LCR position at which the error occurred |

**See Also:**

- "DBA_APPLY_ERROR"
- "USER_APPLY_ERROR"
- *Oracle Database XStream Guide* for information on how to display detailed information about apply errors

# ALL_APPLY_ERROR_MESSAGES

`ALL_APPLY_ERROR_MESSAGES` displays information about the individual messages in an error transaction generated by the apply processes that dequeue messages from queues accessible to the current user.

For XStream inbound servers, each message in an error transaction is an LCR.

**Note:**

- Messages that were spilled from memory to hard disk do not appear in this view.
- This view does not contain information related to XStream outbound servers.

### Related View

`DBA_APPLY_ERROR_MESSAGES` displays information about the individual messages in all of the error transactions generated by all apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `MESSAGE_ID` | `RAW(16)` | | Unique identifier of the message stored in the error queue |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOCAL_TRANSACTION_ID | VARCHAR2(22) | | Local transaction ID for the error transaction |
| TRANSACTION_MESSAGE_NUMBER | NUMBER | | Message number of the message that raised the error. The message number is a sequence number for the messages in the transaction, starting with 1. |
| ERROR_NUMBER | NUMBER | | Error number of the error raised by the transaction. The error number is populated only for the LCR that raised the error. This field is NULL for the other LCRs in the transaction. |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message of the error raised by the transaction. The error message is populated only for the LCR that raised the error. This field is NULL for the other LCRs in the transaction. |
| SOURCE_OBJECT_OWNER | VARCHAR2(128) | | Owner of the object at the source database |
| SOURCE_OBJECT_NAME | VARCHAR2(128) | | Name of the object at the source database |
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the target table. This owner can be different than the SOURCE_OBJECT_OWNER due to user-defined transformations that are possible with apply processing. |
| OBJECT_NAME | VARCHAR2(128) | | Object name of the target table. This object name can be different than the SOURCE_OBJECT_NAME due to user-defined transformations that are possible with apply processing. |
| PRIMARY_KEY | VARCHAR2(4000) | | If this column is populated, it is the primary key of the table row that caused the source transaction to fail at the target. |
| POSITION | RAW(64) | | The LCR position |
| OPERATION | VARCHAR2(100) | | The DML or DDL operation represented in the LCR |
| MESSAGE | CLOB | | The content of the LCR. Content includes column name and value for old and/or new values in DML LCRs. For DDL LCRs, the content is the text of the DDL SQL. |

**See Also:**

- "DBA_APPLY_ERROR_MESSAGES"
- *Oracle Database XStream Guide* for information on managing eager errors encountered by an inbound server

# ALL_APPLY_EXECUTE

ALL_APPLY_EXECUTE displays information about the apply execute actions for the rules visible to the current user.

### Related View

DBA_APPLY_EXECUTE displays information about the apply execute actions for all rules in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| EXECUTE_EVENT | VARCHAR2(2) | | Indicates whether the event satisfying the rule is executed |

**See Also:**

- "DBA_APPLY_EXECUTE"
- *Oracle Streams Concepts and Administration* for information on specifying apply execute actions for apply processes

# ALL_APPLY_INSTANTIATED_GLOBAL

ALL_APPLY_INSTANTIATED_GLOBAL displays information for the current user about databases for which an instantiation SCN has been set.

### Related View

DBA_APPLY_INSTANTIATED_GLOBAL displays information about databases for which an instantiation SCN has been set.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_DATABASE | VARCHAR2(128) | NOT NULL | Name of the database that was instantiated |
| INSTANTIATION_SCN | NUMBER | | Instantiation SCN for the database. Only changes committed after this SCN are applied by an apply process. |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | Database link to which changes are applied. If null, then changes are applied to the local database. |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |

**See Also:**

"DBA_APPLY_INSTANTIATED_GLOBAL"

# ALL_APPLY_INSTANTIATED_OBJECTS

ALL_APPLY_INSTANTIATED_OBJECTS displays information about objects accessible to the current user for which an instantiation SCN has been set.

### Related View

DBA_APPLY_INSTANTIATED_OBJECTS displays information about objects for which an instantiation SCN has been set.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_DATABASE | VARCHAR2(128) | NOT NULL | Name of the database where the object originated |
| SOURCE_OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object at the source database |
| SOURCE_OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object at the source database |
| SOURCE_OBJECT_TYPE | VARCHAR2(11) | | Type of the object at the source database |
| INSTANTIATION_SCN | NUMBER | | Instantiation SCN for the object. Only changes committed after this SCN are applied by an apply process. |
| IGNORE_SCN | NUMBER | | SCN below which the instantiation SCN cannot be set. This value corresponds to the SCN value at the source database at the time when the object was prepared for instantiation. |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | Database link to which changes are applied. If null, then changes are applied to the local database. |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |

**See Also:**

"DBA_APPLY_INSTANTIATED_OBJECTS"

# ALL_APPLY_INSTANTIATED_SCHEMAS

ALL_APPLY_INSTANTIATED_SCHEMAS displays information about schemas accessible to the current user for which an instantiation SCN has been set.

### Related View

DBA_APPLY_INSTANTIATED_SCHEMAS displays information about schemas for which an instantiation SCN has been set.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_DATABASE | VARCHAR2(128) | NOT NULL | Name of the database where the schema originated |
| SOURCE_SCHEMA | VARCHAR2(128) | | Name of the schema at the source database |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INSTANTIATION_SCN | NUMBER | | Instantiation SCN for the schema. Only changes committed after this SCN are applied by an apply process. |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | Database link to which changes are applied. If null, then changes are applied to the local database. |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |

**See Also:**

"DBA_APPLY_INSTANTIATED_SCHEMAS"

# ALL_APPLY_KEY_COLUMNS

ALL_APPLY_KEY_COLUMNS displays information about the substitute key columns for the tables accessible to the current user. Substitute key columns are set using the DBMS_APPLY_ADM.SET_KEY_COLUMNS procedure.

### Related View

DBA_APPLY_KEY_COLUMNS displays information about the substitute key columns for all tables in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object on which substitute key columns are set |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object on which substitute key columns are set |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name of a column specified as a substitute key column |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | Database link to which changes are applied. If null, then changes are applied to the local database. |

**See Also:**

- "DBA_APPLY_KEY_COLUMNS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLY_ADM.SET_KEY_COLUMNS procedure

# ALL_APPLY_PARAMETERS

ALL_APPLY_PARAMETERS displays information about the parameters for the apply processes that dequeue events from queues accessible to the current user.

### Related View

DBA_APPLY_PARAMETERS displays information about the parameters for all apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| APPLY_NAME | VARCHAR2(128) | NOT NULL | Name of the apply process |
| PARAMETER | VARCHAR2(128) | NOT NULL | Name of the parameter |
| VALUE | VARCHAR2(4000) | | Parameter value |
| SET_BY_USER | VARCHAR2(3) | | Indicates whether the parameter value was set by the user (YES) or was not set by the user (NO). If NO for a parameter, then the parameter is set to its default value. If YES for a parameter, then the parameter may or may not be set to its default value. |

**See Also:**

"DBA_APPLY_PARAMETERS"

# ALL_APPLY_PROGRESS

ALL_APPLY_PROGRESS displays information about the progress made by the apply processes that dequeue events from queues accessible to the current user. This view only contains information about captured events. It does not contain information about user-enqueued events.

### Related View

DBA_APPLY_PROGRESS displays information about the progress made by all apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| APPLY_NAME | VARCHAR2(128) | NOT NULL | Name of the apply process |
| SOURCE_DATABASE | VARCHAR2(128) | NOT NULL | Global name of the source database of the changes that are applied by the apply process |
| APPLIED_MESSAGE_NUMBER | NUMBER | NOT NULL | Message number up to which all transactions have definitely been applied. This value is the low watermark for the apply process. That is, messages with a commit message number less than or equal to this message number have definitely been applied, but some messages with a higher commit message number may also have been applied. |
| OLDEST_MESSAGE_NUMBER | NUMBER | NOT NULL | Earliest message number of the transactions currently being dequeued and applied |
| APPLY_TIME | DATE | | Time at which the message with the message number displayed in the |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | APPLIED_MESSAGE_NUMBER column was applied |
| APPLIED_MESSAGE_CREATE_TIME | DATE | | Time at which the message with the message number displayed in the APPLIED_MESSAGE_NUMBER column was created at its source database |
| OLDEST_TRANSACTION_ID | VARCHAR2(128) | | Oldest transaction ID of interest. (useful for detecting long-running or large transactions) |
| SPILL_MESSAGE_NUMBER | NUMBER | | Spill low watermark. Any message with a lower SCN has either been applied or spilled to disk (it will be dequeued from the Streams queue and capture will not need to resend any logical change records (LCRs) with a lower SCN). Spilled messages may not have been applied yet. |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |

**See Also:**

"DBA_APPLY_PROGRESS"

# ALL_APPLY_TABLE_COLUMNS

ALL_APPLY_TABLE_COLUMNS displays, for the tables accessible to the current user, information about the nonkey table columns for which Oracle Streams apply processes do not detect conflicts for updates and deletes.

Conflict detection for nonkey columns can be stopped using the DBMS_APPLY_ADM.COMPARE_OLD_VALUES procedure.

### Related View

DBA_APPLY_TABLE_COLUMNS displays, for all tables in the database, information about the nonkey table columns for which Oracle Streams apply processes do not detect conflicts for updates and deletes.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the table |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column |
| COMPARE_OLD_ON_DELETE | VARCHAR2(3) | | Indicates whether to Compare the old value of the column on deletes (YES) or not (NO) |
| COMPARE_OLD_ON_UPDATE | VARCHAR2(3) | | Indicates whether to Compare the old value of the column on updates (YES) or not (NO) |
| APPLY_DATABASE_LINK | VARCHAR2(128) | | For remote tables, name of the database link pointing to the remote database |

**See Also:**

- "DBA_APPLY_TABLE_COLUMNS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLY_ADM.COMPARE_OLD_VALUES procedure

# ALL_ARGUMENTS

ALL_ARGUMENTS lists the arguments of the functions and procedures that are accessible to the current user.

### Related Views

- DBA_ARGUMENTS lists the arguments of the functions and procedures that are available in the database.
- USER_ARGUMENTS lists the arguments of the functions and procedures that are owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | | Name of the procedure or function |
| PACKAGE_NAME | VARCHAR2(128) | | Name of the package |
| OBJECT_ID | NUMBER | NOT NULL | Object number of the object |
| OVERLOAD | VARCHAR2(40) | | Indicates the *n*th overloading ordered by its appearance in the source; otherwise, it is NULL. |
| SUBPROGRAM_ID | NUMBER | | Unique subprogram identifier |
| ARGUMENT_NAME | VARCHAR2(128) | | If the argument is a scalar type, then the argument name is the name of the argument. A null argument name is used to denote a function return. If the function return or argument is a composite type, this view will have one row for each attribute of the composite type. Attributes are recursively expanded if they are composite.<br>The meanings of ARGUMENT_NAME, POSITION, SEQUENCE, and DATA_LEVEL are interdependent. Together, as a tuple, they represent a node of a flattened tree.<br>ARGUMENT_NAME can refer to any of the following:<br>• Return type, if ARGUMENT_NAME is null and DATA_LEVEL = 0<br>• The argument that appears in the argument list if ARGUMENT_NAME is not null and DATA_LEVEL = 0<br>• Attribute name of the composite type if ARGUMENT_NAME is not null and DATA_LEVEL > 0; |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • A collection element type if ARGUMENT_NAME is null and DATA_LEVEL > 0; |
| POSITION | NUMBER | NOT NULL | If DATA_LEVEL is zero, then this column holds the position of this item in the argument list, or zero for a function return value.If DATA_LEVEL is greater than zero, then this column holds the position of this item with respect to its siblings at the same DATA_LEVEL. So, for a referenced record field, this is the index of the field within the record. For a referenced collection element, this is 1 (because collection elements do not have siblings.) |
| SEQUENCE | NUMBER | NOT NULL | Defines the sequential order of the argument and its attributes. Argument sequence starts from 1. Return type and its recursively expanded (preorder tree walk) attributes will come first, and each argument with its recursively expanded (preorder tree walk) attributes will follow. |
| DATA_LEVEL | NUMBER | NOT NULL | Nesting depth of the argument for composite types |
| DATA_TYPE | VARCHAR2(30) | | Datatype of the argument |
| DEFAULTED | VARCHAR2(1) | | Specifies whether or not the argument is defaulted |
| DEFAULT_VALUE | LONG | | Reserved for future use |
| DEFAULT_LENGTH | NUMBER | | Reserved for future use |
| IN_OUT | VARCHAR2(9) | | Direction of the argument:<br>• IN<br>• OUT<br>• IN/OUT |
| DATA_LENGTH | NUMBER | | Length of the column (in bytes) |
| DATA_PRECISION | NUMBER | | Length in decimal digits (NUMBER) or binary digits (FLOAT) |
| DATA_SCALE | NUMBER | | Digits to the right of the decimal point in a number |
| RADIX | NUMBER | | Argument radix for a number |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Character set name for the argument |
| TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the argument |
| TYPE_NAME | VARCHAR2(128) | | Name of the type of the argument. If the type is a package local type (that is, it is declared in a package specification), then this column displays the name of the package. |
| TYPE_SUBNAME | VARCHAR2(128) | | Relevant only for package local types. Displays the name of the type declared in the package identified in the TYPE_NAME column. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE_LINK | VARCHAR2(128) | | Relevant only for package local types when the package identified in the TYPE_NAME column is a remote package. This column displays the database link used to refer to the remote package. |
| PLS_TYPE | VARCHAR2(128) | | For numeric arguments, the name of the PL/SQL type of the argument. Null otherwise. |
| CHAR_LENGTH | NUMBER | | Character limit for string datatypes |
| CHAR_USED | VARCHAR2(1) | | Indicates whether the byte limit (B) or char limit (C) is official for the string |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**Note:**

To list the procedure names in a package, use the ALL_PROCEDURES view.

**See Also:**

- "DBA_ARGUMENTS"
- "USER_ARGUMENTS"
- "ALL_PROCEDURES" for information about the functions and procedures that are accessible to the current user

# ALL_ASSEMBLIES

ALL_ASSEMBLIES provides information about assemblies accessible to the current user.

### Related Views

- DBA_ASSEMBLIES provides information about all assemblies in the database.
- USER_ASSEMBLIES provides information about all assemblies owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the assembly |
| ASSEMBLY_NAME | VARCHAR2(128) | NOT NULL | Name of the assembly |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_SPEC | VARCHAR2(4000) | | Operating system file specification of the assembly |
| SECURITY_LEVEL | VARCHAR2(10) | | The maximum security level of the assembly |
| IDENTITY | VARCHAR2(4000) | | The identity of the assembly |
| STATUS | VARCHAR2(7) | | Status of the assembly |

**See Also:**

- "DBA_ASSEMBLIES"
- "USER_ASSEMBLIES"

# ALL_ASSOCIATIONS

ALL_ASSOCIATIONS describes user-defined statistics associated with objects accessible to the current user.

### Related Views

- DBA_ASSOCIATIONS describes all user-defined statistics in the database.
- USER_ASSOCIATIONS describes user-defined statistics associated with objects owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object for which the association is being defined |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object for which the association is being defined |
| COLUMN_NAME | VARCHAR2(128) | | Column name in the object for which the association is being defined |
| OBJECT_TYPE | VARCHAR2(9) | | Kind of object with which statistics are being associated: column, type, package or function, indextype, or domain index. |
| STATSTYPE_SCHEMA | VARCHAR2(128) | | Owner of the statistics type |
| STATSTYPE_NAME | VARCHAR2(128) | | Name of statistics type that contains the cost, selectivity or statistics functions |
| DEF_SELECTIVITY | NUMBER | | Default selectivity of the object, if any |
| DEF_CPU_COST | NUMBER | | Default CPU cost of the object, if any |
| DEF_IO_COST | NUMBER | | Default I/O cost of the object, if any |
| DEF_NET_COST | NUMBER | | Default networking cost of the object, if any |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INTERFACE_VERSION | NUMBER | | Identifies the version number of the ODCIStats interface. Value is 1 for statistics type implementing Oracle8*i* 8.1; 0 for types implementing Oracle9*i* 9.0.0. |
| MAINTENANCE_TYPE | VARCHAR2(14) | | Specifies whether the object is system-managed or user-managed |

**See Also:**

- "DBA_ASSOCIATIONS"
- "USER_ASSOCIATIONS"

# ALL_ATTRIBUTE_TRANSFORMATIONS

ALL_ATTRIBUTE_TRANSFORMATIONS displays information about the transformation functions for the transformations accessible to the current user.

### Related Views

- DBA_ATTRIBUTE_TRANSFORMATIONS displays information about the transformation functions for all transformations in the database.
- USER_ATTRIBUTE_TRANSFORMATIONS displays information about the transformation functions for the transformations owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRANSFORMATION_ID | NUMBER | NOT NULL | Unique identifier for the transformation |
| OWNER | VARCHAR2(128) | NOT NULL | Owning user of the transformation |
| NAME | VARCHAR2(128) | NOT NULL | Transformation name |
| FROM_TYPE | VARCHAR2(257) | | Source type name |
| TO_TYPE | VARCHAR2(385) | | Target type name |
| ATTRIBUTE | NUMBER | NOT NULL | Target type attribute number |
| ATTRIBUTE_TRANSFORMATION | VARCHAR2(4000) | | Transformation function for the attribute |

**See Also:**

- "DBA_ATTRIBUTE_TRANSFORMATIONS"
- "USER_ATTRIBUTE_TRANSFORMATIONS"

# ALL_AUDIT_POLICIES

ALL_AUDIT_POLICIES describes the fine-grained auditing policies on the tables and views accessible to the current user.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

### Related Views

- DBA_AUDIT_POLICIES describes all fine-grained auditing policies in the database.
- USER_AUDIT_POLICIES describes the fine-grained auditing policies on the tables and views owned by the current user. This view does not display the OBJECT_SCHEMA column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_SCHEMA | VARCHAR2(128) | | Name of the schema that includes the table or view |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or view |
| POLICY_OWNER | VARCHAR2(128) | | Owner of the policy |
| POLICY_NAME | VARCHAR2(128) | | Name of the policy |
| POLICY_TEXT | VARCHAR2(4000) | | Audit condition |
| POLICY_COLUMN | VARCHAR2(128) | | Relevant column |
| PF_SCHEMA | VARCHAR2(128) | | Owner of the audit handler function |
| PF_PACKAGE | VARCHAR2(128) | | Name of the package containing the audit handler function |
| PF_FUNCTION | VARCHAR2(128) | | Name of the audit handler function |
| ENABLED | VARCHAR2(3) | | Indicates whether the policy is enabled (YES) or disabled (NO) |
| SEL | VARCHAR2(3) | | Indicates whether the policy is applied to queries on the object (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INS | VARCHAR2(3) | | Indicates whether the policy is applied to INSERT statements on the object (YES) or not (NO) |
| UPD | VARCHAR2(3) | | Indicates whether the policy is applied to UPDATE statements on the object (YES) or not (NO) |
| DEL | VARCHAR2(3) | | Indicates whether the policy is applied to DELETE statements on the object (YES) or not (NO) |
| AUDIT_TRAIL | VARCHAR2(12) | | Indicates the audit trail to which the audit records generated by this audit policy will be written:<br>• DB - Audit records are written to DBA_FGA_AUDIT_TRAIL (fine-grained audit trail)<br>• DB+EXTENDED - Audit records are written to DBA_FGA_AUDIT_TRAIL (fine-grained audit trail) and the SQL_TEXT and SQL_BIND columns are populated for this policy<br>• XML - Audit records are written to V $XML_AUDIT_TRAIL (XML audit files)<br>• XML+EXTENDED - Audit records are written to V$XML_AUDIT_TRAIL (XML audit files) and the SQL_TEXT and SQL_BIND columns are populated for this policy |
| POLICY_COLUMN_OPTIONS | VARCHAR2(11) | | Indicates whether all columns in the AUDIT_COLUMN parameter (ALL_COLUMNS) or any of the columns in the AUDIT_COLUMN parameter (ANY_COLUMNS) are considered for triggering fine-grained auditing |

**See Also:**

- "DBA_AUDIT_POLICIES"
- "USER_AUDIT_POLICIES"

# ALL_AUDIT_POLICY_COLUMNS

ALL_AUDIT_POLICY_COLUMNS describes the fine-grained auditing policy columns on the tables and views accessible to the current user.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

### Related Views

- DBA_AUDIT_POLICY_COLUMNS describes all fine-grained auditing policy columns in the database.
- USER_AUDIT_POLICY_COLUMNS describes the fine-grained auditing policy columns on the tables and views owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_SCHEMA | VARCHAR2(128) | | Owner of the table or view |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or view |
| POLICY_NAME | VARCHAR2(128) | | Name of the policy |
| POLICY_COLUMN | VARCHAR2(128) | | Relevant column of the policy |

**See Also:**

- "DBA_AUDIT_POLICY_COLUMNS"
- "USER_AUDIT_POLICY_COLUMNS"

# ALL_AW_PS

ALL_AW_PS describes the page spaces in the analytic workspaces accessible to the current user.

### Related Views

- DBA_AW_PS describes the page spaces in all analytic workspaces in the database.
- USER_AW_PS describes the page spaces in the analytic workspaces owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the analytic workspace |
| AW_NUMBER | NUMBER | NOT NULL | Number of the analytic workspace |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AW_NAME | VARCHAR2(128) | | Name of the analytic workspace |
| PSNUMBER | NUMBER(10) | | Number of the page space |
| GENERATIONS | NUMBER | | Number of active generations in the page space |
| MAXPAGES | NUMBER | | Maximum pages allocated in the page space |

**See Also:**

- "DBA_AW_PS"
- "USER_AW_PS"
- *Oracle OLAP User's Guide* for more information about the OLAP option for Oracle Database

# ALL_AWS

ALL_AWS describes the analytic workspaces accessible to the current user.

### Related Views

- DBA_AWS describes all analytic workspaces in the database.
- USER_AWS describes the analytic workspaces owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the analytic workspace |
| AW_NUMBER | NUMBER | NOT NULL | Number of the analytic workspace |
| AW_NAME | VARCHAR2(128) | | Name of the analytic workspace |
| AW_VERSION | VARCHAR2(4) | | Format version of the analytic workspace:<br>• 9.1<br>• 10.1<br>• 10.2<br>• 11.1 |
| PAGESPACES | NUMBER | | Number of pagespaces in the analytic workspace |
| GENERATIONS | NUMBER | | Number of active generations in the analytic workspace |
| FROZEN | VARCHAR2(6) | | Freeze state of the analytic workspace:<br>• Frozen<br>• NoThaw |

**See Also:**

- "DBA_AWS"
- "USER_AWS"
- *Oracle OLAP User's Guide* for more information about the OLAP option for Oracle Database

# ALL_BASE_TABLE_MVIEWS

ALL_BASE_TABLE_MVIEWS describes the materialized views using materialized view logs accessible to the current user. A materialized view log can be created for a master, base table, or master materialized view. Query this view at the master site or the master materialized view site to show one row for each materialized view using a materialized view log.

### Related Views

- DBA_BASE_TABLE_MVIEWS describes all materialized views using materialized view logs in the database.
- USER_BASE_TABLE_MVIEWS describes the materialized views using materialized view logs owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Schema in which the master table or the master materialized view was created |
| MASTER | VARCHAR2(128) | NOT NULL | Name of the master table or the master materialized view |
| MVIEW_LAST_REFRESH_TIME | DATE | NOT NULL | Date when the materialized view based on the master was last refreshed |
| MVIEW_ID | NUMBER(38) | | Unique identifier of the materialized view that is based on the master |

**See Also:**

- "DBA_BASE_TABLE_MVIEWS"
- "USER_BASE_TABLE_MVIEWS"

# ALL_CAPTURE

ALL_CAPTURE displays information about the capture processes that enqueue the captured changes into queues accessible to the current user.

### Related View

DBA_CAPTURE displays information about all capture processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE_NAME | VARCHAR2(128) | | Name of the capture process |
| QUEUE_NAME | VARCHAR2(128) | | Name of the queue used for staging captured changes |
| QUEUE_OWNER | VARCHAR2(128) | | Owner of the queue used for staging captured changes |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the positive rule set used by the capture process for filtering |
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the positive rule set |
| CAPTURE_USER | VARCHAR2(128) | | Current user who is enqueuing captured messages |
| START_SCN | NUMBER | | System change number (SCN) from which the capture process will start to capture changes. START_SCN is only modified as the result of an ALTER_CAPTURE statement or if the FIRST_SCN moves beyond the existing START_SCN. |
| STATUS | VARCHAR2(8) | | Status of the capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| CAPTURED_SCN | NUMBER | | System change number (SCN) of the last redo log record scanned |
| APPLIED_SCN | NUMBER | | System change number (SCN) of the most recent message dequeued by the relevant apply processes. All changes below this SCN have been dequeued by all apply processes that apply changes captured by this capture process. |
| USE_DATABASE_LINK | VARCHAR2(3) | | Indicates whether the source database name is used as the database link to connect to the source database from the downstream database (YES) or not (NO). If the capture process was created at the source database, then this column will be NULL. |
| FIRST_SCN | NUMBER | | System change number (SCN) from which the capture process can be restarted. FIRST_SCN indicates the lowest SCN to which the capture can be repositioned. |
| SOURCE_DATABASE | VARCHAR2(128) | | Global name of the source database |
| SOURCE_DBID | NUMBER | | Database ID of the source database |
| SOURCE_RESETLOGS_SCN | NUMBER | | Resetlogs system change number (SCN) of the source database |
| SOURCE_RESETLOGS_TIME | NUMBER | | Resetlogs time of the source database |
| LOGMINER_ID | NUMBER | | Session ID of the Oracle LogMiner session associated with the capture process |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NEGATIVE_RULE_SET_NAME | VARCHAR2(128) | | Name of the negative rule set used by the capture process for filtering |
| NEGATIVE_RULE_SET_OWNER | VARCHAR2(128) | | Owner of the negative rule set used by the capture process for filtering |
| MAX_CHECKPOINT_SCN | NUMBER | | System change number (SCN) at which the last checkpoint was taken by the capture process |
| REQUIRED_CHECKPOINT_SCN | NUMBER | | Lowest checkpoint SCN for which the capture process requires redo information.<br>**Note:** This SCN value does not necessarily correspond with a checkpoint SCN value. |
| LOGFILE_ASSIGNMENT | VARCHAR2(8) | | Logfile assignment type for the capture process:<br>• IMPLICIT<br>• EXPLICIT |
| STATUS_CHANGE_TIME | DATE | | Time that the STATUS of the capture process was changed |
| ERROR_NUMBER | NUMBER | | Error number if the capture process was aborted |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message if the capture process was aborted |
| VERSION | VARCHAR2(64) | | Version number of the capture process |
| CAPTURE_TYPE | VARCHAR2(10) | | Type of the capture process:<br>• DOWNSTREAM<br>• LOCAL |
| LAST_ENQUEUED_SCN | NUMBER | | Last enqueued system change number (SCN) |
| CHECKPOINT_RETENTION_TIME | NUMBER | | Checkpoint retention time<br>**Note:** When the checkpoint retention time for a capture process is set to INFINITE, then the value displayed in this column is 4294967295. |
| START_TIME | TIMESTAMP(6) | | Time from which the capture process will start to capture changes. START_TIME is related to START_SCN and can only be modified by an ALTER_CAPTURE statement.<br>You can modify either START_SCN or START_TIME, but not both at the same time. |
| PURPOSE | VARCHAR2(19) | | Purpose of the capture process:<br>• GoldenGate Capture - A capture process configured using Oracle GoldenGate Extract in integrated capture mode<br>• Streams - A capture process in an Oracle Streams configuration<br>• XStream Streams - A capture process in an Oracle Streams configuration with XStream capabilities enabled by the DBMS_XSTREAM_ADM.ENABLE_GG_XSTREAM_FOR_STREAMS procedure |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • XStream Out - A capture process in an XStream Out configuration<br>• AUDIT VAULT - A capture process in an audit vault configuration |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |
| CLIENT_NAME | VARCHAR2(4000) | | Client name of the capture process. This is the outbound name for XStream Out, and the extract name for GoldenGate. |
| CLIENT_STATUS | VARCHAR2(8) | | Status of the client process:<br>• DISABLED - For XStream Out if the outbound server is not running; for GoldenGate if the capture process is not running<br>• DETACHED - For XStream Out if the outbound server is running, but the XStream client application is not attached to it; For GoldenGate if the capture process is running, but the extract process is not attached to it<br>• ATTACHED - For XStream out if the outbound server is running and the XStream client application is attached to it; For GoldenGate if the capture process is running and the extract process is attached to it<br>• ABORTED - For XStream out if the outbound server became disabled because it encountered an error; for GoldenGate if the capture process became disabled because it encountered an error |
| OLDEST_SCN | NUMBER | | Oldest SCN of the transactions currently being processed |
| FILTERED_SCN | NUMBER | | SCN of the low watermark transaction processed |

**See Also:**

- "DBA_CAPTURE"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XSTREAM_ADM.ENABLE_GG_XSTREAM_FOR_STREAMS procedure

# ALL_CAPTURE_EXTRA_ATTRIBUTES

ALL_CAPTURE_EXTRA_ATTRIBUTES displays information about the extra attributes for the capture processes accessible to the current user.

### Related View

DBA_CAPTURE_EXTRA_ATTRIBUTES displays information about the extra attributes for all capture processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the capture process |
| ATTRIBUTE_NAME | VARCHAR2(128) | NOT NULL | Name of the extra attribute |
| INCLUDE | VARCHAR2(3) | | Indicates whether the extra attribute is included (YES) or not (NO) |
| ROW_ATTRIBUTE | VARCHAR2(3) | | Indicates whether the extra attribute is a row LCR attribute (YES) or not (NO) |
| DDL_ATTRIBUTE | VARCHAR2(3) | | Indicates whether the extra attribute is a DDL LCR attribute (YES) or not (NO) |

**See Also:**

"DBA_CAPTURE_EXTRA_ATTRIBUTES"

# ALL_CAPTURE_PARAMETERS

ALL_CAPTURE_PARAMETERS displays information about the parameters for the capture processes that enqueue the captured changes into queues accessible to the current user.

### Related View

DBA_CAPTURE_PARAMETERS displays information about the parameters for all capture processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the capture process |
| PARAMETER | VARCHAR2(128) | NOT NULL | Name of the parameter |
| VALUE | VARCHAR2(4000) | | Parameter value |
| SET_BY_USER | VARCHAR2(3) | | Indicates whether the parameter value was set by the user (YES) or was not set by the user (NO). If NO, then the parameter is set to its default value. If YES, then the parameter may or may not be set to its default value. |
| SOURCE_DATABASE | VARCHAR2(128) | | Global name of the container for which the capture parameter is defined |

**See Also:**

"DBA_CAPTURE_PARAMETERS"

# ALL_CAPTURE_PREPARED_DATABASE

ALL_CAPTURE_PREPARED_DATABASE displays information about when the local database was prepared for instantiation. If the local database was not prepared for instantiation, then this view contains no rows.

### Related View

DBA_CAPTURE_PREPARED_DATABASE displays information about when the local database was prepared for instantiation.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TIMESTAMP | DATE | | Date and time at which the local database was ready to be instantiated |
| SUPPLEMENTAL_LOG_DATA_PK | VARCHAR2(8) | | Status of database-level PRIMARY KEY COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_UI | VARCHAR2(8) | | Status of database-level UNIQUE INDEX COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_FK | VARCHAR2(8) | | Status of database-level FOREIGN KEY COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_ALL | VARCHAR2(8) | | Status of database-level ALL COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |

**See Also:**

- "DBA_CAPTURE_PREPARED_DATABASE"
- *Oracle Streams Concepts and Administration* for more information about the possible values for the supplemental logging columns in this view

# ALL_CAPTURE_PREPARED_SCHEMAS

ALL_CAPTURE_PREPARED_SCHEMAS displays information about the schemas prepared for instantiation that are accessible to the current user at the local database.

### Related View

DBA_CAPTURE_PREPARED_SCHEMAS displays information about all schemas prepared for instantiation at the local database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA_NAME | VARCHAR2(128) | NOT NULL | Name of the schema that is ready to be instantiated |
| TIMESTAMP | DATE | | Date and time at which the schema was ready to be instantiated |
| SUPPLEMENTAL_LOG_DATA_PK | VARCHAR2(8) | | Status of schema-level PRIMARY KEY COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_UI | VARCHAR2(8) | | Status of schema-level UNIQUE INDEX COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_FK | VARCHAR2(8) | | Status of schema-level FOREIGN KEY COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_ALL | VARCHAR2(8) | | Status of schema-level ALL COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |

**See Also:**

- "DBA_CAPTURE_PREPARED_SCHEMAS"
- *Oracle Streams Concepts and Administration* for more information about the possible values for the supplemental logging columns in this view

# ALL_CAPTURE_PREPARED_TABLES

ALL_CAPTURE_PREPARED_TABLES displays information about the tables prepared for instantiation that are accessible to the current user at the local database.

### Related View

DBA_CAPTURE_PREPARED_TABLES displays information about all tables prepared for instantiation at the local database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the table that is ready to be instantiated |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table that is ready to be instantiated |
| SCN | NUMBER | NOT NULL | Smallest system change number (SCN) for which the table can be instantiated |
| TIMESTAMP | DATE | | Date and time at which the table was ready to be instantiated |
| SUPPLEMENTAL_LOG_DATA_PK | VARCHAR2(8) | | Status of table-level PRIMARY KEY COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_UI | VARCHAR2(8) | | Status of table-level UNIQUE INDEX COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_FK | VARCHAR2(8) | | Status of table-level FOREIGN KEY COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |
| SUPPLEMENTAL_LOG_DATA_ALL | VARCHAR2(8) | | Status of table-level ALL COLUMNS supplemental logging:<br>• IMPLICIT<br>• EXPLICIT<br>• NO |

**See Also:**

- "DBA_CAPTURE_PREPARED_TABLES"
- *Oracle Streams Concepts and Administration* for more information about the possible values for the supplemental logging columns in this view

# ALL_CATALOG

ALL_CATALOG displays the tables, clusters, views, synonyms, and sequences accessible to the current user.

### Related Views

- DBA_CATALOG displays all tables, clusters, views, synonyms, and sequences in the entire database.
- USER_CATALOG displays the tables, clusters, views, synonyms, and sequences in the current user's schema. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the TABLE, CLUSTER, VIEW, SYNONYM, SEQUENCE, or UNDEFINED |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the TABLE, CLUSTER, VIEW, SYNONYM, SEQUENCE, or UNDEFINED |
| TABLE_TYPE | VARCHAR2(11) | | Type of the TABLE, CLUSTER, VIEW, SYNONYM, SEQUENCE, or UNDEFINED |

**See Also:**

- "DBA_CATALOG"
- "USER_CATALOG"

# ALL_CLUSTER_HASH_EXPRESSIONS

ALL_CLUSTER_HASH_EXPRESSIONS displays hash functions for all hash clusters accessible to the current user.

### Related Views

- DBA_CLUSTER_HASH_EXPRESSIONS displays hash functions for all hash clusters in the database.
- USER_CLUSTER_HASH_EXPRESSIONS displays hash functions for all hash clusters owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cluster |
| CLUSTER_NAME | VARCHAR2(128) | NOT NULL | Name of the cluster |
| HASH_EXPRESSION | LONG | | Text of the hash function of the hash cluster |

**See Also:**

- "DBA_CLUSTER_HASH_EXPRESSIONS"
- "USER_CLUSTER_HASH_EXPRESSIONS"

# ALL_CLUSTERING_DIMENSIONS

ALL_CLUSTERING_DIMENSIONS describes dimension tables associated with tables with an attribute clustering clause that the user owns or has system privileges for.

### Related Views

- DBA_CLUSTERING_DIMENSIONS describes dimension tables associated with all tables with an attribute clustering clause in the database.

- USER_CLUSTERING_DIMENSIONS describes dimension tables associated with tables with an attribute clustering clause owned by the user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the attribute clustering table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the attribute clustering table |
| DIMENSION_OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension table |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension table |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_CLUSTERING_DIMENSIONS"
- "USER_CLUSTERING_DIMENSIONS"
- The ALTER TABLE section in *Oracle Database SQL Language Reference* for information about using the CLUSTERING clause to create an attribute clustering table
- The CREATE TABLE section in *Oracle Database SQL Language Reference* for information about using the CLUSTERING clause to create an attribute clustering table
- *Oracle Database Data Warehousing Guide* for information about dimension tables
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

# ALL_CLUSTERING_JOINS

ALL_CLUSTERING_JOINS describes joins to the dimension tables associated with tables with an attribute clustering clause the user owns or has system privileges for.

### Related Views

- DBA_CLUSTERING_JOINS describes joins to the dimension tables associated with all tables with an attribute clustering clause in the database.
- USER_CLUSTERING_JOINS describes joins to the dimension tables associated with tables with an attribute clustering clause owned by the user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the attribute clustering table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the attribute clustering table |
| TAB1_OWNER | VARCHAR2(128) | NOT NULL | Table 1 owner of the join |
| TAB1_NAME | VARCHAR2(128) | NOT NULL | Table 1 name of the join |
| TAB1_COLUMN | VARCHAR2(128) | NOT NULL | Table 1 column name of the join |
| TAB2_OWNER | VARCHAR2(128) | NOT NULL | Table 2 owner of the join |
| TAB2_NAME | VARCHAR2(128) | NOT NULL | Table 2 name of the join |
| TAB2_COLUMN | VARCHAR2(128) | NOT NULL | Table 2 column name of the join |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_CLUSTERING_JOINS"
- "USER_CLUSTERING_JOINS"
- The `ALTER TABLE` section in *Oracle Database SQL Language Reference* for information about using the `CLUSTERING` clause to create an attribute clustering table
- The `CREATE TABLE` section in *Oracle Database SQL Language Reference* for information about using the `CLUSTERING` clause to create an attribute clustering table
- *Oracle Database Data Warehousing Guide* for information about dimension tables
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

# ALL_CLUSTERING_KEYS

`ALL_CLUSTERING_KEYS` describes clustering keys for tables with an attribute clustering clause accessible to the user.

### Related Views

- `DBA_CLUSTERING_KEYS` describes clustering keys for all tables with an attribute clustering clause.
- `USER_CLUSTERING_KEYS` describes clustering keys for tables with an attribute clustering clause owned by the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table on which the clustering clause is defined |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table on which the clustering clause is defined |
| DETAIL_OWNER | VARCHAR2(128) | NOT NULL | Owner of the detailed table contributing to the clustering keys |
| DETAIL_NAME | VARCHAR2(128) | NOT NULL | Name of the detailed table contributing to the clustering keys |
| DETAIL_COLUMN | VARCHAR2(128) | NOT NULL | Name of the detail column |
| POSITION | NUMBER | NOT NULL | Position of the column in the clustering clause |
| GROUPID | NUMBER | NOT NULL | Group ID of the column in the clustering clause |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_CLUSTERING_KEYS"
- "USER_CLUSTERING_KEYS"
- The `ALTER TABLE` section in *Oracle Database SQL Language Reference* for information about using the `CLUSTERING` clause to create an attribute clustering table
- The `CREATE TABLE` section in *Oracle Database SQL Language Reference* for information about using the `CLUSTERING` clause to create an attribute clustering table
- *Oracle Database Data Warehousing Guide* for information about dimension tables
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

# ALL_CLUSTERING_TABLES

`ALL_CLUSTERING_TABLES` describes tables with an attribute clustering clause that are accessible to the user.

### Related Views

- `DBA_CLUSTERING_TABLES` describes all the tables with an attribute clustering clause.

- USER_CLUSTERING_TABLES describes the tables with an attribute clustering clause owned by the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| CLUSTERING_TYPE | VARCHAR2(11) | | Clustering type:<br>• INTERLEAVED<br>• LINEAR |
| ON_LOAD | VARCHAR2(3) | | Indicates whether Oracle will cluster data on load (YES) or not (NO) |
| ON_DATAMOVEMENT | VARCHAR2(3) | | Indicates whether Oracle will cluster data on data movement, for example, partition move (YES), or not (NO) |
| VALID | VARCHAR2(3) | | Indicates if clustering is valid (YES) or not (NO).<br>For clustering with dimension tables, it is required that the joins of the fact table to the dimensions is via primary key or unique key on the dimension table. Therefore, dimension join keys must have a valid primary key or unique key constraint. If the primary key or unique key constraint is not valid, then clustering will not occur.<br>If there are no joins in the CLUSTERING clause, then the value of this column is YES. |
| WITH_ZONEMAP | VARCHAR2(3) | | Indicates if a zonemap was created with clustering (YES) or not (NO). |
| LAST_LOAD_CLST | TIMESTAMP(6) | | Last time the clustering occurred on load |
| LAST_DATAMOVE_CLST | TIMESTAMP(6) | | Last time the clustering occurred on data movement, for example, partition move |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_CLUSTERING_TABLES"
- "USER_CLUSTERING_TABLES"
- The `ALTER TABLE` section in *Oracle Database SQL Language Reference* for information about using the `CLUSTERING` clause to create an attribute clustering table
- The `CREATE TABLE` section in *Oracle Database SQL Language Reference* for information about using the `CLUSTERING` clause to create an attribute clustering table
- *Oracle Database Data Warehousing Guide* for information about dimension tables
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

# ALL_CLUSTERS

`ALL_CLUSTERS` describes all clusters accessible to the current user.

### Related Views

- `DBA_CLUSTERS` describes all clusters in the database.
- `USER_CLUSTERS` describes all clusters owned by the current user. This view does not display the `OWNER` column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `OWNER` | `VARCHAR2(128)` | `NOT NULL` | Owner of the cluster |
| `CLUSTER_NAME` | `VARCHAR2(128)` | `NOT NULL` | Name of the cluster |
| `TABLESPACE_NAME` | `VARCHAR2(30)` | `NOT NULL` | Name of the tablespace containing the cluster |
| `PCT_FREE` | `NUMBER` | | Minimum percentage of free space in a block |
| `PCT_USED` | `NUMBER` | | Minimum percentage of used space in a block |
| `KEY_SIZE` | `NUMBER` | | Estimated size of cluster key plus associated rows |
| `INI_TRANS` | `NUMBER` | `NOT NULL` | Initial number of transactions |
| `MAX_TRANS` | `NUMBER` | `NOT NULL` | Maximum number of transactions |
| `INITIAL_EXTENT` | `NUMBER` | | Size of the initial extent in bytes |
| `NEXT_EXTENT` | `NUMBER` | | Size of secondary extents in bytes |
| `MIN_EXTENTS` | `NUMBER` | `NOT NULL` | Minimum number of extents allowed in the segment |
| `MAX_EXTENTS` | `NUMBER` | `NOT NULL` | Maximum number of extents allowed in the segment |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PCT_INCREASE | NUMBER | | Percentage increase in extent size |
| FREELISTS | NUMBER | | Number of process freelists allocated to this segment |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated to this segment |
| AVG_BLOCKS_PER_KEY | NUMBER | | Number of blocks in the table divided by number of cluster keys |
| CLUSTER_TYPE | VARCHAR2(5) | | Type of the cluster:<br>• INDEX - B*-Tree index<br>• HASH - Hash |
| FUNCTION | VARCHAR2(15) | | If the cluster is a hash cluster, the hash function |
| HASHKEYS | NUMBER | | If the cluster is a hash cluster, the number of hash keys (hash buckets) |
| DEGREE | VARCHAR2(10) | | Number of threads per instance for scanning the cluster, or DEFAULT |
| INSTANCES | VARCHAR2(10) | | Number of instances across which the cluster is to be scanned , or DEFAULT |
| CACHE | VARCHAR2(5) | | Indicates whether the cluster is to be cached in the buffer cache (Y) or not (N) |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool to be used for cluster blocks:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for cluster blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for cluster blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| SINGLE_TABLE | VARCHAR2(5) | | Indicates whether this is a single-table cluster (Y) or not (N) |
| DEPENDENCIES | VARCHAR2(8) | | Indicates whether row-level dependency tracking is enabled (ENABLED) or disabled (DISABLED) |

**See Also:**

- "DBA_CLUSTERS"
- "USER_CLUSTERS"

# ALL_CODE_ROLE_PRIVS

ALL_CODE_ROLE_PRIVS describes all the roles that are associated with program units owned or accessible by the current user.

### Related Views

- DBA_CODE_ROLE_PRIVS describes all the roles that are associated with program units in the database.
- USER_CODE_ROLE_PRIVS describes all the roles that are associated with program units owned by current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Username of the owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| OBJECT_TYPE | VARCHAR2(9) | | Type of the object |
| ROLE | VARCHAR2(128) | NOT NULL | The role associated with the object |

**See Also:**

- "DBA_CODE_ROLE_PRIVS"
- "USER_CODE_ROLE_PRIVS"
- *Oracle Database Security Guide* for more information about granting and revoking roles to and from program units

# ALL_COL_COMMENTS

ALL_COL_COMMENTS displays comments on the columns of the tables and views accessible to the current user.

### Related Views

- DBA_COL_COMMENTS displays comments on the columns of all tables and views in the database.
- USER_COL_COMMENTS displays comments on the columns of the tables and views owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column |
| COMMENTS | VARCHAR2(4000) | | Comment on the column |
| ORIGIN_CON_ID | NUMBER | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs<br>• *n*: This value is used for rows containing data that originate in the container with the ID *n* (*n*=1 if the data originates in root) |

**See Also:**

- "DBA_COL_COMMENTS"
- "USER_COL_COMMENTS"

# ALL_COL_PENDING_STATS

ALL_COL_PENDING_STATS describes the pending statistics of the columns accessible to the current user.

### Related Views

- DBA_COL_PENDING_STATS describes the pending statistics of all columns in the database.
- USER_COL_PENDING_STATS describes the pending statistics of the columns owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(32) | | Low value in the column |
| HIGH_VALUE | RAW(32) | | High value in the column |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/NUM_DISTINCT. |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing the column |
| LAST_ANALYZED | DATE | | Most recent date on which the column was analyzed |

**See Also:**

- "DBA_COL_PENDING_STATS"
- "USER_COL_PENDING_STATS"

# ALL_COL_PRIVS

ALL_COL_PRIVS describes the following types of grants:

- Column object grants for which the current user is the object owner, grantor, or grantee
- Column object grants for which an enabled role or PUBLIC is the grantee

### Related Views

- DBA_COL_PRIVS describes all column object grants in the database.
- USER_COL_PRIVS describes the column object grants for which the current user is the object owner, grantor, or grantee.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| TABLE_SCHEMA | VARCHAR2(128) | | Schema of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the column |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |

**See Also:**

- "DBA_COL_PRIVS"
- "USER_COL_PRIVS"

# ALL_COL_PRIVS_MADE

ALL_COL_PRIVS_MADE describes the column object grants for which the current user is the object owner or grantor.

### Related View

USER_COL_PRIVS_MADE describes the column object grants for which the current user is the object owner. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the column |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |

**See Also:**

"USER_COL_PRIVS_MADE"

# ALL_COL_PRIVS_RECD

ALL_COL_PRIVS_RECD describes the following types of grants:

- Column object grants for which the current user is the grantee
- Column object grants for which an enabled role or PUBLIC is the grantee

### Related View

USER_COL_PRIVS_RECD describes the column object grants for which the current user is the grantee. This view does not display the GRANTEE column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the column |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |

**See Also:**

"USER_COL_PRIVS_RECD"

# ALL_COLL_TYPES

ALL_COLL_TYPES describes all named collection types (varrays and nested tables) accessible to the current user.

### Related Views

- DBA_COLL_TYPES describes all named collection types in the database.
- USER_COLL_TYPES describes all named collection types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the collection |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the collection |
| COLL_TYPE | VARCHAR2(128) | NOT NULL | Description of the collection, such as VARYING ARRAY, [nested] TABLE |
| UPPER_BOUND | NUMBER | | For varrays only, maximum size |
| ELEM_TYPE_MOD | VARCHAR2(7) | | Type modifier of the element |
| ELEM_TYPE_OWNER | VARCHAR2(128) | | Owner of the type upon which the collection is based. This value is useful primarily for a user-defined type. |
| ELEM_TYPE_NAME | VARCHAR2(128) | | Name of the data type or user-defined type upon which the collection is based |
| LENGTH | NUMBER | | Length of CHAR elements or maximum length of VARCHAR or VARCHAR2 elements |
| PRECISION | NUMBER | | Decimal precision of NUMBER or DECIMAL elements; binary precision of FLOAT elements |
| SCALE | NUMBER | | Scale of NUMBER or DECIMAL elements |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Name of the character set (CHAR_CS \| NCHAR_CS) |
| ELEM_STORAGE | VARCHAR2(7) | | Obsolete column |
| NULLS_STORED | VARCHAR2(3) | | Obsolete column |
| CHAR_USED | VARCHAR2(1) | | Indicates whether the attribute uses BYTE length semantics (B) or CHAR length semantics (C). For NCHAR and NVARCHAR2 attribute types, this value is always C. |

**See Also:**

- "DBA_COLL_TYPES"
- "USER_COLL_TYPES"

# ALL_CONS_COLUMNS

ALL_CONS_COLUMNS describes columns that are accessible to the current user and that are specified in constraints.

### Related Views

- DBA_CONS_COLUMNS describes all columns in the database that are specified in constraints.
- USER_CONS_COLUMNS describes columns that are owned by the current user and that are specified in constraints.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the constraint definition |
| CONSTRAINT_NAME | VARCHAR2(128) | NOT NULL | Name of the constraint definition |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table with the constraint definition |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column or attribute of the object type column specified in the constraint definition<br><br>**Note:** If you create a constraint on a user-defined REF column, the system creates the constraint on the attributes that make up the REF column. Therefore, the column names displayed in this view are the attribute names, with the REF column name as a prefix, in the following form: "REF_name"."attribute" |
| POSITION | NUMBER | | Original position of the column or attribute in the definition of the object |

**See Also:**

- "DBA_CONS_COLUMNS"
- "USER_CONS_COLUMNS"

# ALL_CONS_OBJ_COLUMNS

ALL_CONS_OBJ_COLUMNS displays information about the types that object columns (or attributes) or collection elements have been constrained to, in the tables accessible to the current user.

### Related Views

- DBA_CONS_OBJ_COLUMNS displays information about the types that object columns (or attributes) or collection elements have been constrained to, in all tables in the database.
- USER_CONS_OBJ_COLUMNS displays information about the types that object columns (or attributes) or collection elements have been constrained to, in the tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table containing the object column or attribute |
| COLUMN_NAME | VARCHAR2(4000) | | Fully qualified name of the object column or attribute |
| CONS_TYPE_OWNER | VARCHAR2(128) | | Owner of the type that the column (or element) is constrained to |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CONS_TYPE_NAME | VARCHAR2(128) | | Name of the type that the column (or element) is constrained to |
| CONS_TYPE_ONLY | VARCHAR2(15) | | Indicates whether the column (or element) is constrained to ONLY type (Y) or not (N) |

**See Also:**

- "DBA_CONS_OBJ_COLUMNS"
- "USER_CONS_OBJ_COLUMNS"

# ALL_CONSTRAINTS

ALL_CONSTRAINTS describes constraint definitions on tables accessible to the current user.

### Related Views

- DBA_CONSTRAINTS describes all constraint definitions in the database.
- USER_CONSTRAINTS describes constraint definitions on tables in the current user's schema.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the constraint definition |
| CONSTRAINT_NAME | VARCHAR2(128) | NOT NULL | Name of the constraint definition |
| CONSTRAINT_TYPE | VARCHAR2(1) | | Type of the constraint definition:<br>• C - Check constraint on a table<br>• P - Primary key<br>• U - Unique key<br>• R - Referential integrity<br>• V - With check option, on a view<br>• O - With read only, on a view<br>• H - Hash expression<br>• F - Constraint that involves a REF column<br>• S - Supplemental logging |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name associated with the table (or view) with the constraint definition |
| SEARCH_CONDITION | LONG | | Text of search condition for a check constraint. This column returns the correct value only when the row originates from the current container. |
| SEARCH_CONDITION_VC | VARCHAR2(4000) | | Text of search condition for a check constraint. This column may truncate the search condition. |
| R_OWNER | VARCHAR2(128) | | Owner of the table referred to in a referential constraint |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| R_CONSTRAINT_NAME | VARCHAR2(128) | | Name of the unique constraint definition for the referenced table |
| DELETE_RULE | VARCHAR2(9) | | Delete rule for a referential constraint:<br>• CASCADE<br>• SET NULL<br>• NO ACTION |
| STATUS | VARCHAR2(8) | | Enforcement status of the constraint:<br>• ENABLED<br>• DISABLED |
| DEFERRABLE | VARCHAR2(14) | | Indicates whether the constraint is deferrable (DEFERRABLE) or not (NOT DEFERRABLE) |
| DEFERRED | VARCHAR2(9) | | Indicates whether the constraint was initially deferred (DEFERRED) or not (IMMEDIATE) |
| VALIDATED | VARCHAR2(13) | | When STATUS = ENABLED, possible values are:<br>• VALIDATED - All data obeys the constraint (that is, the existing data in the table was validated when the constraint was enabled, as well as any subsequent data entered into the table)<br>• NOT VALIDATED - All data may not obey the constraint (that is, the existing data in the table was not validated when the constraint was enabled, but subsequent data entered into the table was validated)<br>When STATUS = DISABLED, possible values are:<br>• VALIDATED - All data obeys the constraint, but the unique index on the constraint has been dropped. This setting is useful in data warehousing environments, but has some restrictions. Refer to *Oracle Database Data Warehousing Guide* for more information on this setting.<br>• NOT VALIDATED - All data may not obey the constraint |
| GENERATED | VARCHAR2(14) | | Indicates whether the name of the constraint is user-generated (USER NAME) or system-generated (GENERATED NAME) |
| BAD | VARCHAR2(3) | | Indicates whether this constraint specifies a century in an ambiguous manner (BAD) or not (NULL). To avoid errors resulting from this ambiguity, rewrite the constraint using the TO_DATE function with a four-digit year.<br>**See Also:** the TO_DATE function in *Oracle Database SQL Language Reference* and *Oracle Database Development Guide* |
| RELY | VARCHAR2(4) | | When VALIDATED = NOT VALIDATED, this column indicates whether the constraint is to be taken into account for query rewrite (RELY) or not (NULL). |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | When VALIDATED = VALIDATED, this column is not meaningful.<br>**See Also:** *constraints* in *Oracle Database SQL Language Reference* |
| LAST_CHANGE | DATE | | When the constraint was last enabled or disabled |
| INDEX_OWNER | VARCHAR2(128) | | Name of the user owning the index |
| INDEX_NAME | VARCHAR2(128) | | Name of the index (only shown for unique and primary-key constraints) |
| INVALID | VARCHAR2(7) | | Indicates whether the constraint is invalid (INVALID) or not (NULL) |
| VIEW_RELATED | VARCHAR2(14) | | Indicates whether the constraint depends on a view (DEPEND ON VIEW) or not (NULL) |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_CONSTRAINTS"
- "USER_CONSTRAINTS"

# ALL_CONTEXT

ALL_CONTEXT describes all context namespaces in the current session for which attributes and values have been specified using the DBMS_SESSION.SET_CONTEXT procedure. This view does not display the TYPE column.

### Related View

DBA_CONTEXT describes all context namespaces defined in the database, regardless whether any attributes have been specified for them using the DBMS_SESSION.SET_CONTEXT procedure.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAMESPACE | VARCHAR2(128) | NOT NULL | Name of the context namespace |
| SCHEMA | VARCHAR2(128) | NOT NULL | Schema name of the designated package that can set attributes using this namespace |
| PACKAGE | VARCHAR2(128) | NOT NULL | Package name of the designated package that can set attributes using this namespace |

**See Also:**

- "DBA_CONTEXT"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SESSION.SET_CONTEXT procedure

# ALL_CREDENTIALS

ALL_CREDENTIALS lists all credentials visible to the user.

### Related Views

- DBA_CREDENTIALS lists all credentials in the database.
- USER_CREDENTIALS lists credentials owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the credential |
| CREDENTIAL_NAME | VARCHAR2(128) | NOT NULL | Name of the credential |
| USERNAME | VARCHAR2(128) | | Name of the user that will be used to log in to the remote database or the remote or local operating system |
| WINDOWS_DOMAIN | VARCHAR2(30) | | For a Windows target, the Windows domain to use when logging in |
| COMMENTS | VARCHAR2(240) | | Comments on the credential |
| ENABLED | VARCHAR2(5) | | Indicates whether this credential is enabled (TRUE) or not (FALSE) |

**Note:**

DBMS_CREDENTIAL lists credentials that can be used to run external procedures, or by DBMS_SCHEDULER for remote or external jobs, or for storing or retrieving files from the operating system.

If a credential is disabled, then any of the actions above that attempts to use the credential will fail.

**See Also:**

- "DBA_CREDENTIALS"
- "USER_CREDENTIALS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_CREDENTIAL package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SCHEDULER package

# ALL_CUBE_ATTR_VISIBILITY

ALL_CUBE_ATTR_VISIBILITY describes the OLAP attributes visible for the dimensions, hierarchies, and levels accessible to the current user.

### Related Views

- DBA_CUBE_ATTR_VISIBILITY describes all OLAP attributes visible for the dimensions, hierarchies, and levels in the database.
- USER_CUBE_ATTR_VISIBILITY describes the OLAP attributes visible for the dimensions, hierarchies, and levels owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | | Name of a cube dimension (such as TIME) |
| ATTRIBUTE_NAME | VARCHAR2(128) | | Name of an attribute of the dimension (such as LONG_DESCRIPTION or END_DATE) |
| HIERARCHY_NAME | VARCHAR2(128) | | Name of a hierarchy for the dimension (such as CALENDAR) |
| LEVEL_NAME | VARCHAR2(128) | | Name of the dimension level (such as MONTH) |
| FROM_TYPE | VARCHAR2(10) | | Identifies the dimension type that the current row derives the attribute visibility from. Possible values:<br>• DIMENSION - Derives the attribute visibility from itself.<br>• HIERARCHY - Derives the attribute visibility from the VisibleAttributes explicitly set on the associated DIMENSION or itself.<br>• DIM_LEVEL - Derives the attribute visibility from the VisibleAttributes explicitly set on the associated DIMENSION or itself.<br>• HIER_LEVEL - Derives the attribute visibility from the VisibleAttributes explicitly set on the associated DIMENSION, HIERARCHY, DIM_LEVEL, or itself. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TO_TYPE | VARCHAR2(10) | | Identifies the dimension type for the current row. Possible values:<br>• DIMENSION - When the TO_TYPE is DIMENSION, then only the DIMENSION_NAME is populated.<br>• HIERARCHY - When the TO_TYPE is HIERARCHY, then only the DIMENSION_NAME and HIERARCHY_NAME are populated.<br>• DIM_LEVEL - When the TO_TYPE is DIM_LEVEL, then only the DIMENSION_NAME and LEVEL_NAME are populated.<br>• HIER_LEVEL - When the TO_TYPE is HIER_LEVEL, then only the HIERARCHY_NAME and LEVEL_NAME are populated. |

**See Also:**

- "DBA_CUBE_ATTR_VISIBILITY"
- "USER_CUBE_ATTR_VISIBILITY"

# ALL_CUBE_ATTRIBUTES

ALL_CUBE_ATTRIBUTES describes the attributes for the OLAP cube dimensions accessible to the current user.

### Related Views

- DBA_CUBE_ATTRIBUTES describes the attributes for all OLAP cube dimensions in the database.
- USER_CUBE_ATTRIBUTES describes the attributes for the OLAP cube dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a cube dimension (such as TIME) |
| ATTRIBUTE_NAME | VARCHAR2(128) | NOT NULL | Name of an attribute of the dimension (such as LONG_DESCRIPTION or END_DATE) |
| ATTRIBUTE_ID | NUMBER | NOT NULL | ID of the attribute of the dimension |
| TARGET_DIMENSION_NAME | VARCHAR2(128) | | Name of the target dimension of the attribute |
| ATTRIBUTE_ROLE | VARCHAR2(17) | | Special role this attribute plays; NULL if none:<br>• SHORT_DESCRIPTION<br>• LONG_DESCRIPTION<br>• DESCRIPTION<br>• TIME_SPAN |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • END_DATE |
| DESCRIPTION | NVARCHAR2(300) | | Description of the attribute in the session language |
| ATTRIBUTE_GROUP_NAME | VARCHAR2(200) | | Name of the attribute group |
| DATA_TYPE | VARCHAR2(106) | | Data type of the attribute, (such as VARCHAR2 or FLOAT) |
| DATA_LENGTH | NUMBER | NOT NULL | Length of a text data type |
| DATA_PRECISION | NUMBER | | Precision of a numeric data type |
| DATA_SCALE | NUMBER | | Scale of a numeric data type |
| CREATE_INDEX | VARCHAR2(3) | | Create index flag of the OLAP attribute. Possible values:<br>• YES: The attribute is represented in the AW as a relation.<br>Setting CreateIndex="True" in the metadata guarantees that it will be represented in the AW as a relation.<br>• NO: The attribute is not represented in the AW as a relation.<br>Setting CreateIndex="False" in the metadata does not guarantee that it will be represented in the AW a a variable; the system will make that determination. |
| IS_MULTI_LINGUAL[1] | VARCHAR2(3) | | Shows the setting for the IsMultiLingual flag of the OLAP Atttribute. Possible values:<br>• YES: The attribute is set as multilingual.<br>Setting IsMultiLingual to True in the metadata means that the attribute can have a value per language instead of a single value.<br>• NO: The attribute is not set as multilingual.<br>Setting IsMultiLingual to False in the metadata means that the attribute has only one value, independent of language. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_CUBE_ATTRIBUTES"
- "USER_CUBE_ATTRIBUTES"

# ALL_CUBE_BUILD_PROCESSES

ALL_CUBE_BUILD_PROCESSES describes the OLAP build processes and maintenance scripts accessible to the current user.

### Related Views

- DBA_CUBE_BUILD_PROCESSES describes all OLAP build processes and maintenance scripts in the database.
- USER_CUBE_BUILD_PROCESSES describes the OLAP build processes and maintenance scripts owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the build process |
| BUILD_PROCESS_NAME | VARCHAR2(128) | NOT NULL | Name of the build process |
| BUILD_PROCESS_ID | NUMBER | NOT NULL | ID of the build process |
| BUILD_PROCESS | CLOB | | Syntax of the build process |
| DESCRIPTION | NVARCHAR2(300) | | Description of the build process in the session language |

**See Also:**

- "DBA_CUBE_BUILD_PROCESSES"
- "USER_CUBE_BUILD_PROCESSES"

# ALL_CUBE_CALCULATED_MEMBERS

ALL_CUBE_CALCULATED_MEMBERS describes the calculated members for the OLAP cube dimensions accessible to the current user.

### Related Views

- DBA_CUBE_CALCULATED_MEMBERS describes the calculated members for all OLAP cube dimensions in the database.
- USER_CUBE_CALCULATED_MEMBERS describes the calculated members for the OLAP cube dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a cube dimension |
| MEMBER_NAME | VARCHAR2(128) | NOT NULL | Name of a calculated member in the cube dimension |
| IS_CUSTOM_AGGREGATE | VARCHAR2(3) | | Indicates whether the calculated member is a custom aggregate (YES) or not (NO) |
| STORAGE_TYPE | VARCHAR2(10) | | Storage type of the calculated member:<br>• DYNAMIC - Value of the member is calculated for a query |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • PRECOMPUTE - Value of the member is calculated and stored during data maintenance |
| EXPRESSION | CLOB | | Expression used to generate the value of the calculated member |

**See Also:**

- "DBA_CUBE_CALCULATED_MEMBERS"
- "USER_CUBE_CALCULATED_MEMBERS"

# ALL_CUBE_DIM_LEVELS

ALL_CUBE_DIM_LEVELS describes the OLAP dimension levels accessible to the current user.

### Related Views

- DBA_CUBE_DIM_LEVELS describes all OLAP dimension levels in the database.
- USER_CUBE_DIM_LEVELS describes the OLAP dimension levels owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a dimension, such as CUSTOMER |
| LEVEL_NAME | VARCHAR2(128) | NOT NULL | Name of a level in the dimension, such as WAREHOUSE |
| LEVEL_ID | NUMBER | NOT NULL | ID of the dimension level |
| DESCRIPTION | NVARCHAR2(300) | | Description of the dimension level in the session language |

**See Also:**

- "DBA_CUBE_DIM_LEVELS"
- "USER_CUBE_DIM_LEVELS"

# ALL_CUBE_DIM_MODELS

ALL_CUBE_DIM_MODELS describes the models for the OLAP dimensions accessible to the current user.

### Related Views

- DBA_CUBE_DIM_MODELS describes the models for all OLAP dimensions in the database.
- USER_CUBE_DIM_MODELS describes the models for the OLAP dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a cube dimension |
| MODEL_NAME | VARCHAR2(128) | NOT NULL | Name of a model for the cube dimension |
| MODEL_ID | NUMBER | NOT NULL | ID of the model |
| DESCRIPTION | NVARCHAR2(300) | | Description of the model in the session language |

**See Also:**

- "DBA_CUBE_DIM_MODELS"
- "USER_CUBE_DIM_MODELS"

# ALL_CUBE_DIM_VIEW_COLUMNS

ALL_CUBE_DIM_VIEW_COLUMNS describes the columns of the relational views of the OLAP cube dimensions accessible to the current user.

### Related Views

- DBA_CUBE_DIM_VIEW_COLUMNS describes the columns of the relational views of all OLAP cube dimensions in the database.
- USER_CUBE_DIM_VIEW_COLUMNS describes the columns of the relational views of the OLAP cube dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | | Name of a cube dimension, such as PRODUCT |
| VIEW_NAME | VARCHAR2(128) | | Name of a view of the dimension, such as PRODUCT_VIEW |
| COLUMN_NAME | VARCHAR2(128) | | Name of a column in the view, such as LONG_DESCRIPTION or WAREHOUSE_ID |
| COLUMN_TYPE | VARCHAR2(11) | | Type of the column:<br>• KEY - A key of the dimension view (that is, the dimension value itself) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • LEVEL_NAME - Name of the level (if any) corresponding to a row in the view<br>• DIM_ORDER - A column by which the results may be ordered (if present)<br>• MEMBER_TYPE<br>• ATTRIBUTE - An attribute owned by the dimension |
| OBJECT_NAME | VARCHAR2(128) | | Name of the level or attribute represented in the column, such as LONG_DESCRIPTION or WAREHOUSE_ID |

**See Also:**

- "DBA_CUBE_DIM_VIEW_COLUMNS"
- "USER_CUBE_DIM_VIEW_COLUMNS"

# ALL_CUBE_DIM_VIEWS

ALL_CUBE_DIM_VIEWS describes the relational views of the OLAP dimensions accessible to the current user.

### Related Views

- DBA_CUBE_DIM_VIEWS describes the relational views of all OLAP dimensions in the database.
- USER_CUBE_DIM_VIEWS describes the relational views of the OLAP dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a cube dimension, such as PRODUCT |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of a view of the cube dimension, such as PRODUCT_VIEW |

**See Also:**

- "DBA_CUBE_DIM_VIEWS"
- "USER_CUBE_DIM_VIEWS"

# ALL_CUBE_DIMENSIONALITY

ALL_CUBE_DIMENSIONALITY describes the dimension order for the OLAP cubes accessible to the current user.

### Related Views

- DBA_CUBE_DIMENSIONALITY describes the dimension order for all OLAP cubes in the database.
- USER_CUBE_DIMENSIONALITY describes the dimension order for the OLAP cubes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube |
| CUBE_NAME | VARCHAR2(128) | NOT NULL | Name of a cube, such as UNITS_CUBE |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a dimension of the cube, such as PRODUCT |
| DIMENSIONALITY_NAME | VARCHAR2(200) | | The name of a dimensionality of the cube. For example, a cube dimensioned by the PRODUCT dimension can have a product dimension named PRODUCT_DIM. |
| DIMENSIONALITY_ID | NUMBER | NOT NULL | ID of the cube dimensionality |
| ORDER_NUM | NUMBER | NOT NULL | Order number of the dimension in the cube |
| IS_SPARSE | NUMBER | | Indicates whether the dimension is sparse in the cube (1) or not sparse (0) |
| ET_ATTR_PREFIX | VARCHAR2(200) | | Specifies the prefix that will be added to the column names in the Materialized Views to ensure uniqueness. If the user does not specify an ET_ATTR_PREFIX for any dimensions in a cube, then they default in the pattern D1_, D2_, and so on. |

**See Also:**

- "DBA_CUBE_DIMENSIONALITY"
- "USER_CUBE_DIMENSIONALITY"

## ALL_CUBE_DIMENSIONS

ALL_CUBE_DIMENSIONS describes the OLAP cube dimensions accessible to the current user.

### Related Views

- DBA_CUBE_DIMENSIONS describes all OLAP cube dimensions in the database.
- USER_CUBE_DIMENSIONS describes the OLAP cube dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube dimension |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a cube dimension, such as TIME |
| DIMENSION_ID | NUMBER | NOT NULL | ID of the cube dimension |
| DIMENSION_TYPE | VARCHAR2(17) | | Type of the OLAP cube dimension:<br>• STANDARD<br>• TIME |
| AW_NAME | VARCHAR2(128) | | Name of the analytic workspace that contains the cube dimension, such as GLOBAL |
| DEFAULT_HIERARCHY_NAME | VARCHAR2(128) | | Name of the default hierarchy for the cube dimension, such as FISCAL |
| DESCRIPTION | NVARCHAR2(300) | | Description of the cube dimension in the session language |
| HIERARCHY_CONSISTENCY_RULE | VARCHAR2(200) | | Hierarchy consistency rule of the OLAP cube dimension. Possible values:<br>• CONSISTENT<br>• STAR_CONSISTENT<br>• SOLVE_CONSISTENT |
| ADD_UNIQUE_KEY_PREFIX | VARCHAR2(3) | | Add_Unique_Key_Prefix flag of the OLAP cube dimension. Possible values:<br>• YES: This is the value if AddUniqueKeyPrefix="True" was set in the metadata.<br>This tells the system to add the level name prefix to the dimension members. This should be done when a dimension member can have the same value across different levels, for example, New York (state) and New York (city).<br>• NO: This is the value if AddUniqueKeyPrefix="True" was not set in the metadata. |
| CUSTOM_ORDER | CLOB | | The textual representation of the sort orderby clause used to load dimension members into the AW |

**See Also:**

- "DBA_CUBE_DIMENSIONS"
- "USER_CUBE_DIMENSIONS"

# ALL_CUBE_HIER_LEVELS

ALL_CUBE_HIER_LEVELS describes the hierarchy levels for the OLAP cube dimensions accessible to the current user.

### Related Views

- DBA_CUBE_HIER_LEVELS describes the hierarchy levels for all OLAP cube dimensions in the database.
- USER_CUBE_HIER_LEVELS describes the hierarchy levels for the OLAP cube dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a cube dimension, such as TIME |
| HIERARCHY_NAME | VARCHAR2(128) | NOT NULL | Name of a hierarchy for the dimension, such as CALENDAR |
| LEVEL_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension level, such as MONTH |
| HIERARCHY_LEVEL_ID | NUMBER | NOT NULL | ID of the hierarchy level |
| ORDER_NUM | NUMBER | NOT NULL | Order number of the level within the hierarchy; 0 is the top level |
| DESCRIPTION | NVARCHAR2(300) | | Description of the level in the session language |

**See Also:**

- "DBA_CUBE_HIER_LEVELS"
- "USER_CUBE_HIER_LEVELS"

# ALL_CUBE_HIER_VIEW_COLUMNS

ALL_CUBE_HIER_VIEW_COLUMNS describes the columns of the relational hierarchy views of the OLAP cube dimensions accessible to the current user.

### Related Views

- DBA_CUBE_HIER_VIEW_COLUMNS describes the columns of the relational hierarchy views of all OLAP cube dimensions in the database.
- USER_CUBE_HIER_VIEW_COLUMNS describes the columns of the relational hierarchy views of the OLAP cube dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | | Name of a cube dimension, such as TIME |
| HIERARCHY_NAME | VARCHAR2(128) | | Name of a hierarchy for the cube dimension, such as CALENDAR |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| VIEW_NAME | VARCHAR2(128) | | Name of a view of the hierarchy, such as TIME_CALENDAR_VIEW |
| COLUMN_NAME | VARCHAR2(128) | | Name of a column in the view, such as CALENDAR_QUARTER or PARENT |
| COLUMN_TYPE | VARCHAR2(11) | | Type of the column:<br>• KEY - A key of the hierarchy view (that is, the hierarchy value itself)<br>• PARENT - Dimension value of the parent of the current row in the view (or NULL if no parent)<br>• LEVEL_NAME - Name of the level (if any) corresponding to a row in the view<br>• DEPTH - Depth in the hierarchy tree of the current row in the view<br>• HIER_ORDER - A column by which the results may be ordered (if present)<br>• MEMBER_TYPE<br>• ATTRIBUTE - An attribute owned by the hierarchy<br>• LEVEL - One of the level columns comprising the hierarchy |
| OBJECT_NAME | VARCHAR2(128) | | Name of a level or attribute for the dimension |

**See Also:**

- "DBA_CUBE_HIER_VIEW_COLUMNS"
- "USER_CUBE_HIER_VIEW_COLUMNS"

# ALL_CUBE_HIER_VIEWS

ALL_CUBE_HIER_VIEWS describes the hierarchies for the OLAP cube dimensions accessible to the current user.

### Related Views

- DBA_CUBE_HIER_VIEWS describes the hierarchies for all OLAP cube dimensions in the database.
- USER_CUBE_HIER_VIEWS describes the hierarchies for the OLAP cube dimensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a cube dimension, such as TIME |
| HIERARCHY_NAME | VARCHAR2(128) | NOT NULL | Name of a hierarchy for the cube dimension, such as CALENDAR |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of a view of the hierarchy, such as TIME_CALENDAR_VIEW |

**See Also:**

- "DBA_CUBE_HIER_VIEWS"
- "USER_CUBE_HIER_VIEWS"

# ALL_CUBE_HIERARCHIES

ALL_CUBE_HIERARCHIES describes the OLAP dimension hierarchies accessible to the current user.

### Related Views

- DBA_CUBE_HIERARCHIES describes all OLAP dimension hierarchies in the database.
- USER_CUBE_HIERARCHIES describes the OLAP dimension hierarchies owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of a dimension, such as TIME |
| HIERARCHY_NAME | VARCHAR2(128) | NOT NULL | Name of a hierarchy for the dimension, such as CALENDAR |
| HIERARCHY_ID | NUMBER | NOT NULL | ID of the hierarchy |
| HIERARCHY_TYPE | VARCHAR2(5) | | Type of the hierarchy:<br>• LEVEL<br>• VALUE |
| DESCRIPTION | NVARCHAR2(300) | | Description of the hierarchy in the session language |
| IS_RAGGED | NUMBER | | Indicates whether ragged hierarchies are permitted in subsequent builds. User dimensions that are enabled for materialized views and Time dimensions are set to 0. Builds then check the data for ragged hierarchies and fail if one is detected. When User dimensions are set to 1, the builds do not check for ragged hierarchies. |
| IS_SKIP_LEVEL | NUMBER | | Indicates whether skip-level hierarchies are permitted in subsequent builds. User dimensions that are enabled for materialized views and Time dimensions are set to 0. Builds then check the data for skip-level hierarchies and fail if one is detected. When User dimensions are set to 1, the builds do not check for skip-level hierarchies. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REFRESH_MVIEW_NAME | VARCHAR2(200) | | Name of the Refresh Materialized View associated with the hierarchy |
| CUSTOM_ORDER | CLOB | | The textual representation of the sort orderby clause used to load dimension members of the hierarchy into the AW |

**See Also:**

- "DBA_CUBE_HIERARCHIES"
- "USER_CUBE_HIERARCHIES"

# ALL_CUBE_MEASURES

ALL_CUBE_MEASURES describes the measures for the OLAP cubes accessible to the current user.

### Related Views

- DBA_CUBE_MEASURES describes the measures for all OLAP cubes in the database.
- USER_CUBE_MEASURES describes the measures for the OLAP cubes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube |
| CUBE_NAME | VARCHAR2(128) | NOT NULL | Name of a cube, such as UNITS_CUBE |
| MEASURE_NAME | VARCHAR2(128) | NOT NULL | Name of a measure in the cube, such as SALES |
| MEASURE_ID | NUMBER | NOTNULL | ID of a measure |
| OVERRIDE_SOLVE_SPEC | CLOB | | Syntax text for the measure's consistent solve specification that overrides that of its cube |
| MEASURE_TYPE | VARCHAR2(7) | | Type of the OLAP measure:<br>• BASE - Base measures store the data<br>• DERIVED - Derived measures calculate the data from base measures; also called calculated measures |
| EXPRESSION | CLOB | | Expression that provides the values of the measure |
| DESCRIPTION | NVARCHAR2(300) | | Description of the measure in the session language |
| DATA_TYPE | VARCHAR2(106) | | Data type of the measure, such as NUMBER |
| DATA_LENGTH | NUMBER | NOT NULL | Length of a character data type |
| DATA_PRECISION | NUMBER | | Precision of a numeric data type |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DATA_SCALE | NUMBER | | Scale of a numeric data type |
| LOOP_VAR_OVERRIDE | VARCHAR2(200) | | The value that overrides the $LOOP_VAR property of the OLAP derived measure |
| LOOP_DENSE_OVERRIDE | VARCHAR2(200) | | The value that overrides the $LOOP_DENSE property of the OLAP derived measure |
| LOOP_TYPE | VARCHAR2(200) | | The $LOOP_TYPE property of the OLAP derived measure.<br>Possible values:<br>• INNER<br>• OUTER<br>• DENSE |

**See Also:**

- "DBA_CUBE_MEASURES"
- "USER_CUBE_MEASURES"

# ALL_CUBE_NAMED_BUILD_SPECS

ALL_CUBE_NAMED_BUILD_SPECS describes the OLAP cube named build specifications in the database that are accessible by the user.

### Related Views

- DBA_CUBE_NAMED_BUILD_SPECS describes the OLAP cube named build specifications in the database.
- USER_CUBE_NAMED_BUILD_SPECS describes the OLAP cube named build specifications in the database that are owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the OLAP named build specification |
| CUBE_NAME | VARCHAR2(128) | NOT NULL | Name of the OLAP cube |
| NAMED_BUILD_SPEC | CLOB | | Name of the OLAP cube named build specification |

**See Also:**

- "DBA_CUBE_NAMED_BUILD_SPECS"
- "USER_CUBE_NAMED_BUILD_SPECS"

# ALL_CUBE_SUB_PARTITION_LEVELS

ALL_CUBE_SUB_PARTITION_LEVELS describes the OLAP secondary partition levels in the database that are accessible by the user.

### Related Views

- DBA_CUBE_SUB_PARTITION_LEVELS describes the OLAP secondary partition levels in the database.
- USER_CUBE_SUB_PARTITION_LEVELS describes the OLAP secondary partition levels in the database that are owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the OLAP secondary partition level |
| CUBE_NAME | VARCHAR2(128) | | Name of the OLAP cube |
| SUB_PARTITION_LEVEL_NAME | VARCHAR2(200) | | Name of the secondary partition level of the OLAP cube |
| PRECOMPUTE_PERCENT | NUMBER | | Precompute percent of the secondary partition level of the OLAP cube |
| PARTITION_DIMENSION_NAME | VARCHAR2(128) | | Name of the cube dimension for which there is a secondary partition level on the OLAP cube |
| PARTITION_HIERARCHY_NAME | VARCHAR2(128) | | Name of the hierarchy for which there is a secondary partition level on the OLAP cube |
| PARTITION_LEVEL_NAME | VARCHAR2(128) | | Name of the hierarchy level for which there is a secondary partition level on the OLAP cube |
| SUB_PARTITION_LEVEL_ORDER | NUMBER | | Order number of the secondary partition level on the OLAP cube |

**See Also:**

- "DBA_CUBE_SUB_PARTITION_LEVELS"
- "USER_CUBE_SUB_PARTITION_LEVELS"

# ALL_CUBE_VIEW_COLUMNS

ALL_CUBE_VIEW_COLUMNS describes the columns of the relational views of the OLAP cubes accessible to the current user.

### Related Views

- DBA_CUBE_VIEW_COLUMNS describes the columns of relational views of all OLAP cubes in the database.

- USER_CUBE_VIEW_COLUMNS describes the columns of relational views of OLAP cubes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the cube |
| CUBE_NAME | VARCHAR2(128) | | Name of a cube, such as UNITS_CUBE |
| VIEW_NAME | VARCHAR2(128) | | Name of a view of the cube, such as PRODUCT_VIEW |
| COLUMN_NAME | VARCHAR2(128) | | Name of a column in the view, such as DIM_KEY or LEVEL_NAME |
| COLUMN_TYPE | VARCHAR2(7) | | Type of the column:<br>• MEASURE<br>• KEY |
| OBJECT_NAME | VARCHAR2(128) | | Name of the measure or dimension represented in the column |

**See Also:**

- "DBA_CUBE_VIEW_COLUMNS"
- "USER_CUBE_VIEW_COLUMNS"

# ALL_CUBE_VIEWS

ALL_CUBE_VIEWS describes the relational views of the OLAP cubes accessible to the current user.

### Related Views

- DBA_CUBE_VIEWS describes the relational views of all OLAP cubes in the database.
- USER_CUBE_VIEWS describes the relational views of the OLAP cubes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube |
| CUBE_NAME | VARCHAR2(128) | NOT NULL | Name of a cube, such as UNITS_CUBE |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of a view of the cube, such as UNITS_CUBE_VIEW |

**See Also:**

- "DBA_CUBE_VIEWS"
- "USER_CUBE_VIEWS"

# ALL_CUBES

ALL_CUBES describes the OLAP cubes accessible to the current user.

### Related Views

- DBA_CUBES describes all OLAP cubes in the database.
- USER_CUBES describes the OLAP cubes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the cube |
| CUBE_NAME | VARCHAR2(128) | NOT NULL | Name of a cube, such as UNITS_CUBE |
| CUBE_ID | NUMBER | NOT NULL | ID of a cube |
| AW_NAME | VARCHAR2(128) | | Name of the analytic workspace that contains the cube, such as GLOBAL |
| CONSISTENT_SOLVE_SPEC | CLOB | | Default aggregation rules for the cube |
| DESCRIPTION | NVARCHAR2(300) | | Description of the cube in the session language |
| SPARSE_TYPE | VARCHAR2(200) | | Text value indicating the type of sparsity for the OLAP cube |
| PRECOMPUTE_CONDITION | CLOB | | Condition syntax representing the precompute condition of the OLAP cube |
| PRECOMPUTE_PERCENT | NUMBER | | Percentage of aggregate data values that are calculated and stored during data maintenance. If the cube is partitioned, then this percentage is for the bottom partitions. |
| PRECOMPUTE_PERCENT_TOP | NUMBER | | Percentage of aggregate data values in the top partition that are calculated and stored during data maintenance |
| PARTITION_DIMENSION_NAME | VARCHAR2(128) | | Name of the dimension used to partition the cube, such as TIME |
| PARTITION_HIERARCHY_NAME | VARCHAR2(128) | | Name of the dimension hierarchy used to partition the cube, such as CALENDAR |
| PARTITION_LEVEL_NAME | VARCHAR2(128) | | Name of the level used to partition the cube, such as QUARTER |
| REFRESH_MVIEW_NAME | VARCHAR2(200) | | Name of the refresh materialized view for the OLAP cube |
| REWRITE_MVIEW_NAME | VARCHAR2(200) | | Name of the rewrite materialized view for the OLAP cube |
| DEFAULT_BUILD_SPEC | CLOB | | The default build specification for the OLAP cube |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MEASURE_STORAGE | VARCHAR2(200) | | The measure storage for the OLAP cube. Possible values:<br>• INDEPENDENT<br>• SHARED |
| SQL_CUBE_STORAGE_TYPE | CLOB | | The SQL cube storage type for the OLAP cube. This value represents a SQL data type. |
| CUBE_STORAGE_TYPE | VARCHAR2(200) | | The cube storage type for the OLAP cube. This value represents a DML data type. |

**See Also:**

- "DBA_CUBES"
- "USER_CUBES"

# ALL_DB_LINKS

ALL_DB_LINKS describes the database links accessible to the current user.

### Related Views

- DBA_DB_LINKS describes all database links in the database.
- USER_DB_LINKS describes the database links owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the database link |
| DB_LINK | VARCHAR2(128) | NOT NULL | Name of the database link |
| USERNAME | VARCHAR2(128) | | Name of the user when logging in |
| HOST | VARCHAR2(2000) | | Oracle Net connect string |
| CREATED | DATE | NOT NULL | Creation time of the database link |

**See Also:**

- "DBA_DB_LINKS"
- "USER_DB_LINKS"

# ALL_DEF_AUDIT_OPTS

ALL_DEF_AUDIT_OPTS contains default object-auditing options that will be applied when objects are created. The output for each column takes one of the following forms:

- `-/-`: no default auditing
- `S/-`: auditing whenever successful
- `-/S`: auditing whenever not successful

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ALT | VARCHAR2(3) | | Auditing `ALTER` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| AUD | VARCHAR2(3) | | Auditing `AUDIT` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| COM | VARCHAR2(3) | | Auditing `COMMENT` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| DEL | VARCHAR2(3) | | Auditing `DELETE` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| GRA | VARCHAR2(3) | | Auditing `GRANT` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| IND | VARCHAR2(3) | | Auditing `INDEX` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| INS | VARCHAR2(3) | | Auditing `INSERT` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| LOC | VARCHAR2(3) | | Auditing `LOCK` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| REN | VARCHAR2(3) | | Auditing `RENAME` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| SEL | VARCHAR2(3) | | Auditing `SELECT` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| UPD | VARCHAR2(3) | | Auditing `UPDATE` `WHENEVER SUCCESSFUL` / `UNSUCCESSFUL` |
| REF | CHAR(3) | | This column is obsolete and maintained for backward compatibility. The value of this column is always `-/-` |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXE | VARCHAR2(3) | | Auditing EXECUTE WHENEVER SUCCESSFUL / UNSUCCESSFUL |
| FBK | VARCHAR2(3) | | Auditing FLASHBACK WHENEVER SUCCESSFUL / UNSUCCESSFUL |
| REA | VARCHAR2(3) | | Auditing READ WHENEVER SUCCESSFUL / UNSUCCESSFUL |

# ALL_DEPENDENCIES

ALL_DEPENDENCIES describes dependencies between procedures, packages, functions, package bodies, and triggers accessible to the current user, including dependencies on views created without any database links. This view does not display the SCHEMAID column.

### Related Views

- DBA_DEPENDENCIES describes all dependencies between objects in the database. This view does not display the SCHEMAID column.
- USER_DEPENDENCIES describes dependencies between objects in the current user's schema. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| TYPE | VARCHAR2(18) | | Type of the object |
| REFERENCED_OWNER | VARCHAR2(128) | | Owner of the referenced object (remote owner if remote object) |
| REFERENCED_NAME | VARCHAR2(128) | | Name of the referenced object |
| REFERENCED_TYPE | VARCHAR2(18) | | Type of the referenced object |
| REFERENCED_LINK_NAME | VARCHAR2(128) | | Name of the link to the parent object (if remote) |
| DEPENDENCY_TYPE | VARCHAR2(4) | | Indicates whether the dependency is a REF dependency (REF) or not (HARD) |

**See Also:**

- "DBA_DEPENDENCIES"
- "USER_DEPENDENCIES"

# ALL_DEQUEUE_QUEUES

ALL_DEQUEUE_QUEUES describes all queues on which the current user has dequeue privileges. If the user has any Advanced Queuing system privileges, such as DEQUEUE ANY QUEUE or MANAGE ANY QUEUE, then this view describes all queues in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue |
| NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| QUEUE_TABLE | VARCHAR2(128) | NOT NULL | Name of the table in which the queue data resides |
| QID | NUMBER | NOT NULL | Object number of the queue |
| QUEUE_TYPE | VARCHAR2(20) | | Type of the queue:<br>• EXCEPTION_QUEUE<br>• NORMAL_QUEUE |
| MAX_RETRIES | NUMBER | | Maximum number of retries allowed when dequeuing from the queue |
| RETRY_DELAY | NUMBER | | Time interval between retries |
| ENQUEUE_ENABLED | VARCHAR2(7) | | Indicates whether the queue is enabled for enqueue ( YES ) or not ( NO ) |
| DEQUEUE_ENABLED | VARCHAR2(7) | | Indicates whether the queue is enabled for dequeue ( YES ) or not ( NO ) |
| RETENTION | VARCHAR2(40) | | Time interval that processed messages are retained in the queue, or FOREVER |
| USER_COMMENT | VARCHAR2(50) | | User-specified comment |
| NETWORK_NAME | VARCHAR2(512) | | Network name of the queue service |
| SHARDED | VARCHAR2(5) | | TRUE if the queue is sharded, FALSE otherwise |

# ALL_DIM_ATTRIBUTES

ALL_DIM_ATTRIBUTES describes the relationship between a dimension level and a functionally dependent column. The level columns and the dependent column must be in the same table.

### Related Views

- DBA_DIM_ATTRIBUTES describes all such dimension relationships in the database.
- USER_DIM_ATTRIBUTES describes all such dimension attributes in the current user's schema.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension |
| ATTRIBUTE_NAME | VARCHAR2(128) | | Name of the attribute |
| LEVEL_NAME | VARCHAR2(128) | | Name of the hierarchy level |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Dependent column name |
| INFERRED | CHAR(1) | | Indicates whether the attribute is inferred from a JOIN KEY specification (Y) or not (N) |

**See Also:**

- "DBA_DIM_ATTRIBUTES"
- "USER_DIM_ATTRIBUTES"

# ALL_DIM_CHILD_OF

ALL_DIM_CHILD_OF describes hierarchical relationships of 1 to *n* between the pairs of levels in the dimensions accessible to the current user.

### Related Views

- DBA_DIM_CHILD_OF describes all such hierarchical relationships in the database.
- USER_DIM_CHILD_OF describes all such hierarchical attributes in the current user's schema.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension |
| HIERARCHY_NAME | VARCHAR2(128) | | Hierarchy name |
| POSITION | NUMBER | NOT NULL | Hierarchical position within this hierarchy, position 1 being the most detailed |
| CHILD_LEVEL_NAME | VARCHAR2(128) | | Child side of 1:n relationship |
| JOIN_KEY_ID | VARCHAR2(40) | | If non-null, then the child joins to the parent |
| PARENT_LEVEL_NAME | VARCHAR2(128) | | Parent side of 1:n relationship in relation to the CHILD_LEVEL_NAME |

**See Also:**

- "DBA_DIM_CHILD_OF"
- "USER_DIM_CHILD_OF"

# ALL_DIM_HIERARCHIES

ALL_DIM_HIERARCHIES describes all dimension hierarchies accessible to the current user.

### Related Views

- DBA_DIM_HIERARCHIES describes all such hierarchies in the database.
- USER_DIM_HIERARCHIES describes all such hierarchies owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension |
| HIERARCHY_NAME | VARCHAR2(128) | | Hierarchy name |

**See Also:**

- "DBA_DIM_HIERARCHIES"
- "USER_DIM_HIERARCHIES"

# ALL_DIM_JOIN_KEY

ALL_DIM_JOIN_KEY describes the joins between two dimension tables that are accessible to the current user. The join is always specified between a parent dimension level column and a child column.

### Related Views

- DBA_DIM_JOIN_KEY describes all such joins in the database.
- USER_DIM_JOIN_KEY describes all such joins owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension |
| DIM_KEY_ID | NUMBER | NOT NULL | Join key ID (unique within a dimension) |
| LEVEL_NAME | VARCHAR2(128) | | Name of the hierarchy level |
| KEY_POSITION | NUMBER | NOT NULL | Ordinal position of the key column within the level |
| HIERARCHY_NAME | VARCHAR2(128) | | Name of the hierarchy |
| CHILD_JOIN_OWNER | VARCHAR2(128) | NOT NULL | Owner of the join column table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CHILD_JOIN_TABLE | VARCHAR2(128) | NOT NULL | Name of the join column table |
| CHILD_JOIN_COLUMN | VARCHAR2(128) | NOT NULL | Name of the join column |
| CHILD_LEVEL_NAME | VARCHAR2(128) | | Name of the child hierarchy level of the join key |

**See Also:**

- "DBA_DIM_JOIN_KEY"
- "USER_DIM_JOIN_KEY"

# ALL_DIM_LEVEL_KEY

ALL_DIM_LEVEL_KEY describes a column of a dimension level accessible to the current user. The position of a column within a level is specified by KEY_POSITION.

### Related Views

- DBA_DIM_LEVEL_KEY describes all columns of dimension levels in the database.
- USER_DIM_LEVEL_KEY describes all columns of dimension levels owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension |
| LEVEL_NAME | VARCHAR2(128) | | Name of the hierarchy level |
| KEY_POSITION | NUMBER | NOT NULL | Ordinal position of the key column within the level |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the key column |

**See Also:**

- "DBA_DIM_LEVEL_KEY"
- "USER_DIM_LEVEL_KEY"

# ALL_DIM_LEVELS

ALL_DIM_LEVELS describes the dimension levels accessible to the current user. All columns of a dimension level must come from the same relation.

### Related Views

- DBA_DIM_LEVELS describes all dimension levels in the database.

- USER_DIM_LEVELS describes the levels of all dimensions owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension |
| LEVEL_NAME | VARCHAR2(128) | | Unique within a dimension |
| NUM_COLUMNS | NUMBER | | Number of columns in the level definition |
| DETAILOBJ_OWNER | VARCHAR2(128) | NOT NULL | Owner of the detail object that the keys of this level come from |
| DETAILOBJ_NAME | VARCHAR2(128) | NOT NULL | Name of the table that the keys of this level come from |
| SKIP_WHEN_NULL | VARCHAR2(1) | | Indicates whether the level is declared with the SKIP WHEN NULL clause (Y) or not (N) |

**See Also:**

- "DBA_DIM_LEVELS"
- "USER_DIM_LEVELS"

# ALL_DIMENSIONS

ALL_DIMENSIONS describes the dimension objects accessible to the current user.

### Related Views

- DBA_DIMENSIONS describes all dimensions in the database.
- USER_DIMENSIONS describes the dimensions in the current user's schema.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension |
| DIMENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension |
| INVALID | VARCHAR2(1) | | Indicates whether the dimension is invalid (Y) or valid (N) |
| COMPILE_STATE | VARCHAR2(13) | | Compile status of the dimension:<br>• INVALID<br>• NEEDS_COMPILE<br>• ERROR |
| REVISION | NUMBER | | Dimension revision level |

**See Also:**

- "DBA_DIMENSIONS"
- "USER_DIMENSIONS"

# ALL_DIRECTORIES

ALL_DIRECTORIES describes all directories accessible to the current user.

### Related View

DBA_DIRECTORIES describes all directories in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the directory (always SYS) |
| DIRECTORY_NAME | VARCHAR2(128) | NOT NULL | Name of the directory |
| DIRECTORY_PATH | VARCHAR2(4000) | | Operating system pathname for the directory |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

"DBA_DIRECTORIES"

# ALL_EDITION_COMMENTS

ALL_EDITION_COMMENTS describes the comments on the editions accessible to the current user.

### Related View

DBA_EDITION_COMMENTS describes the comments on all editions in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EDITION_NAME | VARCHAR2(128) | NOT NULL | Name of the edition |
| COMMENTS | VARCHAR2(4000) | | Edition comments |

**See Also:**

"DBA_EDITION_COMMENTS"

# ALL_EDITIONING_VIEW_COLS

ALL_EDITIONING_VIEW_COLS describes the relationship between the columns of the editioning views accessible to the current user and the table columns to which they map.

### Related Views

- DBA_EDITIONING_VIEW_COLS describes the relationship between the columns of all editioning views in the database and the table columns to which they map.
- USER_EDITIONING_VIEW_COLS describes the relationship between the columns of the editioning views owned by the current user and the table columns to which they map. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of an editioning view |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of an editioning view |
| VIEW_COLUMN_ID | NUMBER | NOT NULL | Column number within the editioning view |
| VIEW_COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column in the editioning view |
| TABLE_COLUMN_ID | NUMBER | NOT NULL | Column number of a table column to which this editioning view column maps |
| TABLE_COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of a table column to which this editioning view column maps |

**See Also:**

- "DBA_EDITIONING_VIEW_COLS"
- "USER_EDITIONING_VIEW_COLS"

# ALL_EDITIONING_VIEW_COLS_AE

ALL_EDITIONING_VIEW_COLS_AE describes the relationship between the columns of the editioning views (across all editions) accessible to the current user and the table columns to which they map.

### Related Views

- DBA_EDITIONING_VIEW_COLS_AE describes the relationship between the columns of all editioning views (across all editions) in the database and the table columns to which they map.
- USER_EDITIONING_VIEW_COLS_AE describes the relationship between the columns of the editioning views (across all editions) owned by the current user and the table columns to which they map. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of an editioning view |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of an editioning view |
| VIEW_COLUMN_ID | NUMBER | NOT NULL | Column number within the editioning view |
| VIEW_COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column in the editioning view |
| TABLE_COLUMN_ID | NUMBER | NOT NULL | Column number of a table column to which this editioning view column maps |
| TABLE_COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of a table column to which this editioning view column maps |
| EDITION_NAME | VARCHAR2(128) | | Name of the application edition where the editioning view is defined |

**See Also:**

- "DBA_EDITIONING_VIEW_COLS_AE"
- "USER_EDITIONING_VIEW_COLS_AE"

# ALL_EDITIONING_VIEWS

ALL_EDITIONING_VIEWS describes the editioning views accessible to the current user.

### Related Views

- DBA_EDITIONING_VIEWS describes all editioning views in the database.
- USER_EDITIONING_VIEWS describes the editioning views owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of an editioning view |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of an editioning view |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of an editioning view's base table |

**See Also:**

- "DBA_EDITIONING_VIEWS"
- "USER_EDITIONING_VIEWS"

# ALL_EDITIONING_VIEWS_AE

ALL_EDITIONING_VIEWS_AE describes the editioning views (across all editions) accessible to the current user.

### Related Views

- DBA_EDITIONING_VIEWS_AE describes all editioning views (across all editions) in the database.
- USER_EDITIONING_VIEWS_AE describes the editioning views (across all editions) owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of an editioning view |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of an editioning view |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of an editioning view's base table |
| EDITION_NAME | VARCHAR2(128) | | Name of the application edition where the editioning view is defined |

**See Also:**

- "DBA_EDITIONING_VIEWS_AE"
- "USER_EDITIONING_VIEWS_AE"

# ALL_EDITIONS

ALL_EDITIONS describes the editions accessible to the current user.

### Related View

DBA_EDITIONS describes all editions in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EDITION_NAME | VARCHAR2(128) | NOT NULL | Name of the edition |
| PARENT_EDITION_NAME | VARCHAR2(128) | | Name of the parent edition for this edition |
| USABLE | VARCHAR2(3) | | Indicates whether the edition is usable (YES) or unusable (NO) |

**See Also:**

- "DBA_EDITIONS"
- *Oracle Database Development Guide* for more information about editions

# ALL_ENCRYPTED_COLUMNS

ALL_ENCRYPTED_COLUMNS displays encryption algorithm information for the encrypted columns in the tables accessible to the current user.

### Related Views

- DBA_ENCRYPTED_COLUMNS displays encryption algorithm information for all encrypted columns in the database.
- USER_ENCRYPTED_COLUMNS displays encryption algorithm information for the encrypted columns in the tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column |
| ENCRYPTION_ALG | VARCHAR2(29) | | Encryption algorithm used to protect secrecy of data in this column:<br>• 3 Key Triple DES 168 bits key<br>• AES 128 bits key<br>• AES 192 bits key<br>• AES 256 bits key |
| SALT | VARCHAR2(3) | | Indicates whether the column is encrypted with SALT (YES) or not (NO) |
| INTEGRITY_ALG | VARCHAR2(12) | | Integrity algorithm used for the column:<br>• SHA-1<br>• NOMAC |

**See Also:**

- "DBA_ENCRYPTED_COLUMNS"
- "USER_ENCRYPTED_COLUMNS"

# ALL_ERROR_TRANSLATIONS

ALL_ERROR_TRANSLATIONS describes all error translations accessible to the user.

### Related Views

- DBA_ERROR_TRANSLATIONS describes all error translations in the database.
- USER_ERROR_TRANSLATIONS describes all error translations owned by the user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the SQL translation profile |
| PROFILE_NAME | VARCHAR2(128) | NOT NULL | Name of the SQL translation profile |
| ERROR_CODE | NUMBER | NOT NULL | The Oracle error code |
| TRANSLATED_CODE | NUMBER | | The translated error code |
| TRANSLATED_SQLSTATE | VARCHAR2(5) | | The translated SQLSTATE |
| ENABLED | VARCHAR2(5) | | TRUE if the translation is enabled, FALSE otherwise. |
| REGISTRATION_TIME | TIMESTAMP(6) | | Time the translation was registered |
| COMMENTS | VARCHAR2(4000) | | Comment on the translation |

**See Also:**

- "DBA_ERROR_TRANSLATIONS"
- "USER_ERROR_TRANSLATIONS"

# ALL_ERRORS

ALL_ERRORS describes the current errors on the stored objects accessible to the current user.

### Related Views

- DBA_ERRORS describes the current errors on all stored objects in the database.
- USER_ERRORS describes the current errors on the stored objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| TYPE | VARCHAR2(12) | | Type of the object:<br>• VIEW<br>• PROCEDURE<br>• FUNCTION<br>• PACKAGE<br>• PACKAGE BODY<br>• TRIGGER<br>• TYPE<br>• TYPE BODY<br>• LIBRARY<br>• JAVA SOURCE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • JAVA CLASS<br>• DIMENSION |
| SEQUENCE | NUMBER | NOT NULL | Sequence number (for ordering purposes) |
| LINE | NUMBER | NOT NULL | Line number at which the error occurred |
| POSITION | NUMBER | NOT NULL | Position in the line at which the error occurred |
| TEXT | VARCHAR2(4000) | NOT NULL | Text of the error |
| ATTRIBUTE | VARCHAR2(9) | | Indicates whether the error is an error (ERROR) or a warning (WARNING) |
| MESSAGE_NUMBER | NUMBER | | Numeric error number (without any prefix) |

**See Also:**

- "DBA_ERRORS"
- "USER_ERRORS"

# ALL_ERRORS_AE

ALL_ERRORS_AE describes the current errors on the stored objects (across all editions) accessible to the current user.

### Related Views

- DBA_ERRORS_AE describes the current errors on all stored objects (across all editions) in the database.
- USER_ERRORS_AE describes the current errors on the stored objects (across all editions) owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| TYPE | VARCHAR2(12) | | Type of the object:<br>• TYPE<br>• TYPE BODY<br>• VIEW<br>• PROCEDURE<br>• FUNCTION<br>• PACKAGE<br>• PACKAGE BODY<br>• TRIGGER<br>• JAVA SOURCE<br>• JAVA CLASS |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEQUENCE | NUMBER | NOT NULL | Sequence number (for ordering purposes) |
| LINE | NUMBER | NOT NULL | Line number at which this error occurred |
| POSITION | NUMBER | NOT NULL | Position in the line at which this error occurred |
| TEXT | VARCHAR2(4000) | NOT NULL | Text of the error |
| ATTRIBUTE | VARCHAR2(9) | | Indicates whether the error is an error (ERROR) or a warning (WARNING) |
| MESSAGE_NUMBER | NUMBER | | Numeric error number (without any prefix) |
| EDITION_NAME | VARCHAR2(128) | | Name of the edition in which the object is actual |

**See Also:**

- "DBA_ERRORS_AE"
- "USER_ERRORS_AE"

# ALL_EVALUATION_CONTEXT_TABLES

ALL_EVALUATION_CONTEXT_TABLES describes the tables in the rule evaluation contexts accessible to the current user.

### Related Views

- DBA_EVALUATION_CONTEXT_TABLES describes the tables in all rule evaluation contexts in the database.
- USER_EVALUATION_CONTEXT_TABLES describes the tables in the rule evaluation contexts owned by the current user. This view does not display the EVALUATION_CONTEXT_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVALUATION_CONTEXT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the evaluation context |
| EVALUATION_CONTEXT_NAME | VARCHAR2(128) | NOT NULL | Name of the evaluation context |
| TABLE_ALIAS | VARCHAR2(128) | | Alias for a table in the evaluation context |
| TABLE_NAME | VARCHAR2(4000) | | Name of the table referred to by the table alias |

**See Also:**

- "DBA_EVALUATION_CONTEXT_TABLES"
- "USER_EVALUATION_CONTEXT_TABLES"

# ALL_EVALUATION_CONTEXT_VARS

ALL_EVALUATION_CONTEXT_VARS describes the variables in the rule evaluation contexts accessible to the current user.

### Related Views

- DBA_EVALUATION_CONTEXT_VARS describes the variables in all rule evaluation contexts in the database.
- USER_EVALUATION_CONTEXT_VARS describes the variables in the rule evaluation contexts owned by the current user. This view does not display the EVALUATION_CONTEXT_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVALUATION_CONTEXT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the evaluation context |
| EVALUATION_CONTEXT_NAME | VARCHAR2(128) | NOT NULL | Name of the evaluation context |
| VARIABLE_NAME | VARCHAR2(128) | | Name of a variable in the evaluation context |
| VARIABLE_TYPE | VARCHAR2(4000) | | Datatype of the variable |
| VARIABLE_VALUE_FUNCTION | VARCHAR2(4000) | | Function used to retrieve the value of the variable; NULL for variables that are not implicit |
| VARIABLE_METHOD_FUNCTION | VARCHAR2(228) | | Function used to retrieve the result of method invocation on the variable. Such a function can speed up evaluation, if there are many simple rules that invoke the method on the variable. |

**See Also:**

- "DBA_EVALUATION_CONTEXT_VARS"
- "USER_EVALUATION_CONTEXT_VARS"

# ALL_EVALUATION_CONTEXTS

ALL_EVALUATION_CONTEXTS describes the rule evaluation contexts accessible to the current user.

### Related Views

- DBA_EVALUATION_CONTEXTS describes all rule evaluation contexts in the database.
- USER_EVALUATION_CONTEXTS describes the rule evaluation contexts owned by the current user. This view does not display the EVALUATION_CONTEXT_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVALUATION_CONTEXT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the evaluation context |
| EVALUATION_CONTEXT_NAME | VARCHAR2(128) | NOT NULL | Name of the evaluation context |
| EVALUATION_FUNCTION | VARCHAR2(4000) | | Evaluation function associated with the evaluation context, if any |
| EVALUATION_CONTEXT_COMMENT | VARCHAR2(4000) | | Comment specified with the evaluation context, if any |

**See Also:**

- "DBA_EVALUATION_CONTEXTS"
- "USER_EVALUATION_CONTEXTS"

## ALL_EXTERNAL_LOCATIONS

ALL_EXTERNAL_LOCATIONS describes the locations (data sources) of the external tables accessible to the current user.

### Related Views

- DBA_EXTERNAL_LOCATIONS describes the locations (data sources) of all external tables in the database.
- USER_EXTERNAL_LOCATIONS describes the locations (data sources) of the external tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the external table location |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the corresponding external table |
| LOCATION | VARCHAR2(4000) | | External table location clause |
| DIRECTORY_OWNER | CHAR(3) | | Owner of the directory containing the external table location |
| DIRECTORY_NAME | VARCHAR2(128) | | Name of the directory containing the external table location |

**See Also:**

- "DBA_EXTERNAL_LOCATIONS"
- "USER_EXTERNAL_LOCATIONS"

# ALL_EXTERNAL_TABLES

ALL_EXTERNAL_TABLES describes the external tables accessible to the current user.

### Related Views

- DBA_EXTERNAL_TABLES describes all external tables in the database.
- USER_EXTERNAL_TABLES describes the external tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the external table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the external table |
| TYPE_OWNER | CHAR(3) | | Owner of the implementation type for the external table access driver |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the implementation type for the external table access driver |
| DEFAULT_DIRECTORY_OWNER | CHAR(3) | | Owner of the default directory for the external table |
| DEFAULT_DIRECTORY_NAME | VARCHAR2(128) | NOT NULL | Name of the default directory for the external table |
| REJECT_LIMIT | VARCHAR2(40) | | Reject limit for the external table, or UNLIMITED |
| ACCESS_TYPE | VARCHAR2(7) | | Type of access parameters for the external table:<br>• BLOB<br>• CLOB |
| ACCESS_PARAMETERS | CLOB | | Access parameters for the external table |
| PROPERTY | VARCHAR2(10) | | Property of the external table:<br>• REFERENCED - Referenced columns<br>• ALL - All columns |

**See Also:**

- "DBA_EXTERNAL_TABLES"
- "USER_EXTERNAL_TABLES"

# ALL_FILE_GROUP_EXPORT_INFO

ALL_FILE_GROUP_EXPORT_INFO shows export-related information for each version accessible to the current user. There will only be information in this view for versions that have a valid Data Pump dump file.

### Related Views

- DBA_FILE_GROUP_EXPORT_INFO shows export-related information for each version in the database that has a valid Data Pump dump file.
- USER_FILE_GROUP_EXPORT_INFO shows export-related information for all file groups owned by the current user. This view does not display the FILE_GROUP_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_GROUP_OWNER | VARCHAR2(128) | NOT NULL | Owner of the file group |
| FILE_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the file group |
| VERSION_NAME | VARCHAR2(128) | NOT NULL | User-specified name for the version |
| VERSION | NUMBER | NOT NULL | Internal version number |
| EXPORT_VERSION | VARCHAR2(128) | NOT NULL | Version of exported objects |
| PLATFORM_NAME | VARCHAR2(101) | NOT NULL | Platform on which the export was performed |
| EXPORT_TIME | DATE | NOT NULL | Time at which the export job was performed |
| EXPORT_SCN | NUMBER | | SCN of the export job |
| SOURCE_GLOBAL_NAME | VARCHAR2(128) | | Global name of the exporting database |

**See Also:**

- "DBA_FILE_GROUP_EXPORT_INFO"
- "USER_FILE_GROUP_EXPORT_INFO"

# ALL_FILE_GROUP_FILES

ALL_FILE_GROUP_FILES shows the file set for each versioned file group accessible to the current user.

### Related Views

- DBA_FILE_GROUP_FILES shows the file set for each versioned group in the database.
- USER_FILE_GROUP_FILES shows the file set for each versioned group owned by the current user. This view does not display the FILE_GROUP_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_GROUP_OWNER | VARCHAR2(128) | NOT NULL | Owner of the file group |
| FILE_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the file group |
| VERSION_NAME | VARCHAR2(128) | NOT NULL | Name of the version to which the file belongs |
| VERSION | NUMBER | NOT NULL | Internal version number of the file group version to which the file belongs |
| FILE_NAME | VARCHAR2(512) | NOT NULL | Name of the file |
| FILE_DIRECTORY | VARCHAR2(128) | NOT NULL | Directory object for the directory where the file is stored |
| FILE_TYPE | VARCHAR2(32) | | User-specified file type |
| FILE_SIZE | NUMBER | | Size of the file |
| FILE_BLOCK_SIZE | NUMBER | | Block size for the file |
| COMMENTS | VARCHAR2(4000) | | Comments about the file group |

**See Also:**

- "DBA_FILE_GROUP_FILES"
- "USER_FILE_GROUP_FILES"

# ALL_FILE_GROUP_TABLES

ALL_FILE_GROUP_TABLES shows information about the tables accessible to the current user that can be imported using the file set.

### Related Views

- DBA_FILE_GROUP_TABLES shows information about all the tables in the database that can be imported using the file set.
- USER_FILE_GROUP_TABLES shows information about tables owned by the current user that can be imported using the file set. This view does not display the FILE_GROUP_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_GROUP_OWNER | VARCHAR2(128) | NOT NULL | Owner of the file group |
| FILE_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the file group |
| VERSION_NAME | VARCHAR2(128) | NOT NULL | Version of the file group that contains the table |
| VERSION | NUMBER | NOT NULL | Internal version number |
| OWNER | VARCHAR2(128) | NOT NULL | Schema to which the table belongs |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace to which the table belongs |
| SCN | NUMBER | | SCN at which the table was exported (available only for Streams-prepared tables) |

**See Also:**

- "DBA_FILE_GROUP_TABLES"
- "USER_FILE_GROUP_TABLES"

# ALL_FILE_GROUP_TABLESPACES

ALL_FILE_GROUP_TABLESPACES shows information about the transportable tablespaces present (partially or completely) in the file groups accessible to the current user (when the file groups contain dump files).

### Related Views

- DBA_FILE_GROUP_TABLESPACES shows information about the transportable tablespaces present (partially or completely) in all file groups in the database (when the file groups contain dump files).
- USER_FILE_GROUP_TABLESPACES shows information about the transportable tablespaces present (partially or completely) in the file groups owned by the current user (when the file groups contain dump files). This view does not display the FILE_GROUP_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_GROUP_OWNER | VARCHAR2(128) | NOT NULL | Owner of the file group |
| FILE_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the file group |
| VERSION_NAME | VARCHAR2(128) | NOT NULL | Version of the file group that contains the tablespace |
| VERSION | NUMBER | NOT NULL | Internal version number |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace |

**See Also:**

- "DBA_FILE_GROUP_TABLESPACES"
- "USER_FILE_GROUP_TABLESPACES"

# ALL_FILE_GROUP_VERSIONS

ALL_FILE_GROUP_VERSIONS shows top-level version information for the file groups accessible to the current user.

### Related Views

- DBA_FILE_GROUP_VERSIONS shows top-level version information for all file groups in the database.
- USER_FILE_GROUP_VERSIONS shows top-level version information for all file groups owned by the current user. This view does not display the FILE_GROUP_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_GROUP_OWNER | VARCHAR2(128) | NOT NULL | Owner of the file group |
| FILE_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the file group |
| VERSION_NAME | VARCHAR2(128) | NOT NULL | User-specified name for the version |
| VERSION | NUMBER | NOT NULL | Internal version number |
| CREATOR | VARCHAR2(128) | NOT NULL | User who created the version |
| CREATED | TIMESTAMP(6) WITH TIME ZONE | NOT NULL | Time at which the version was created |
| COMMENTS | VARCHAR2(4000) | | Comments about the file group |
| DEFAULT_DIRECTORY | VARCHAR2(128) | | Default directory object for this version, if specified |

**See Also:**

- "DBA_FILE_GROUP_VERSIONS"
- "USER_FILE_GROUP_VERSIONS"

# ALL_FILE_GROUPS

ALL_FILE_GROUPS shows top-level metadata about the file groups accessible to the current user.

### Related Views

- DBA_FILE_GROUPS shows top-level metadata about all file groups in the database.
- USER_FILE_GROUPS shows top-level metadata about file groups owned by the current user. This view does not display the FILE_GROUP_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_GROUP_OWNER | VARCHAR2(128) | NOT NULL | Owner of the file group |
| FILE_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the file group |
| KEEP_FILES | VARCHAR2(1) | NOT NULL | A value of Y or N to indicate whether or not files should be deleted when a version is purged |
| MIN_VERSIONS | NUMBER | NOT NULL | Autopurge should not drop a version if this condition will become violated |
| MAX_VERSIONS | NUMBER | NOT NULL | Autopurge will drop the oldest version when this condition is violated |
| RETENTION_DAYS | NUMBER | NOT NULL | Autopurge will drop versions older than this if doing so does not violate MIN_VERSIONS |
| CREATED | TIMESTAMP(6) WITH TIME ZONE | NOT NULL | Time at which the file group was created |
| COMMENTS | VARCHAR2(4000) | | Comments about the file group |
| DEFAULT_DIRECTORY | VARCHAR2(128) | | Name of the default directory object |

**See Also:**

- "DBA_FILE_GROUPS"
- "USER_FILE_GROUPS"

# ALL_GG_INBOUND_PROGRESS

ALL_GG_INBOUND_PROGRESS displays information about the progress made by the GoldenGate inbound servers accessible to the current user.

### Related View

DBA_GG_INBOUND_PROGRESS displays information about the progress made by all GoldenGate inbound servers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SERVER_NAME | VARCHAR2(128) | NOT NULL | Name of the inbound server |
| PROCESSED_LOW_POSITION | VARCHAR2(4000) | | Position of the processed low transaction |
| APPLIED_LOW_POSITION | VARCHAR2(4000) | | All messages with commit position less than this value have been applied.<br>This column should be used to view the progress of the GoldenGate apply. This column will hold an Oracle SCN numeric value in text format for an Oracle source database. For a non-Oracle source database, this column will hold the apply |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | low position in GoldenGate CSN text format for that specific source database. |
| APPLIED_HIGH_POSITION | VARCHAR2(4000) | | Highest commit position of a transaction that has been applied |
| SPILL_POSITION | VARCHAR2(4000) | | Position of the spill low watermark of the transactions currently being applied |
| OLDEST_POSITION | VARCHAR2(4000) | | Earliest position of the transactions currently being applied |
| APPLIED_LOW_SCN | NUMBER | NOT NULL | All SCN below or equal to this number have been successfully applied. This column is not applicable for GoldenGate replication since the source database may be non-Oracle. |
| APPLIED_TIME | DATE | | Time at which the APPLIED_MESSAGE_NUMBER message was applied |
| APPLIED_MESSAGE_CREATE_TIME | DATE | | Time at which the APPLIED_MESSAGE_NUMBER message was created |
| SOURCE_DATABASE | VARCHAR2(128) | | Database where the transaction originated |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database where all transactions originated |
| LOGBSN[1] | VARCHAR2(4000) | | Log BSN value from the GoldenGate trail file |

1 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DBA_GG_INBOUND_PROGRESS"

# ALL_GOLDENGATE_INBOUND

ALL_GOLDENGATE_INBOUND displays information about the GoldenGate inbound servers accessible to the current user.

### Related View

DBA_GOLDENGATE_INBOUND displays information about all GoldenGate inbound servers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REPLICAT_NAME | VARCHAR2(4000) | | The name of the replicat group created from GGSCI using GoldenGate |
| SERVER_NAME | VARCHAR2(128) | NOT NULL | Name of the inbound server |
| APPLY_USER | VARCHAR2(128) | | Name of the user who can connect to the inbound server and apply messages |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_COMMENT | VARCHAR2(4000) | | User comment |
| CREATE_DATE | TIMESTAMP(6) | | Date when inbound server was created |
| STATUS | VARCHAR2(8) | | Status of the inbound server:<br>• DISABLED - The inbound server is not running.<br>• DETACHED - The inbound server is running, but the GoldenGate client application is not attached to it.<br>• ATTACHED - The inbound server is running, and the GoldenGate client application is attached to it.<br>• ABORTED - The inbound server became disabled because it encountered an error. |

**See Also:**

"DBA_GOLDENGATE_INBOUND"

# ALL_GOLDENGATE_PRIVILEGES

ALL_GOLDENGATE_PRIVILEGES displays details about Oracle GoldenGate privileges for the user.

Oracle GoldenGate privileges are granted using the DBMS_GOLDENGATE_AUTH package.

### Related Views

- DBA_GOLDENGATE_PRIVILEGES displays details about Oracle GoldenGate privileges for all users who have been granted Oracle GoldenGate privileges.
- USER_GOLDENGATE_PRIVILEGES displays details about Oracle GoldenGate privileges. This view does not display the USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user that is granted the privilege |
| PRIVILEGE_TYPE | VARCHAR2(7) | | Type of privilege granted:<br>• APPLY<br>• CAPTURE<br>• *: Both APPLY and CAPTURE |
| GRANT_SELECT_PRIVILEGES | VARCHAR2(3) | | Shows whether the set of privileges granted to the administrator make the administrator a full privilege administrator or a minimum privilege administrator:<br>• YES: The administrator has the SELECT_CATALOG_ROLE role and other privileges, is considered a full privilege administrator, and can manage any Oracle GoldenGate configuration. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NO: The administrator is considered a minimum privilege administrator, and can only manage Oracle GoldenGate configurations where the apply_user or capture_user (based on the PRIVILEGE_TYPE column) matches the username. |
| CREATE_TIME | TIMESTAMP(6) | | Timestamp for the granted privilege |

**See Also:**

- "DBA_GOLDENGATE_PRIVILEGES"
- "USER_GOLDENGATE_PRIVILEGES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_GOLDENGATE_AUTH package

# ALL_GOLDENGATE_RULES

ALL_GOLDENGATE_RULES displays information about the GoldenGate rules accessible to the current user.

### Related View

DBA_GOLDENGATE_RULES displays information about all GoldenGate server rules in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPONENT_NAME | VARCHAR2(128) | | Name of the GoldenGate process |
| COMPONENT_TYPE | VARCHAR2(12) | | Type of the GoldenGate process:<br>• CAPTURE<br>• APPLY |
| COMPONENT_RULE_TYPE | VARCHAR2(9) | | For global, schema or table rules, the GoldenGate type of the rule:<br>• TABLE<br>• SCHEMA<br>• GLOBAL |
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the rule set |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the rule set |
| RULE_SET_TYPE | CHAR(8) | | Type of the rule set:<br>• POSITIVE<br>• NEGATIVE |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_TYPE | VARCHAR2(9) | | For global, schema or table rules, the type of the rule:<br>• DML<br>• DDL |
| RULE_CONDITION | CLOB | | Current rule condition |
| SCHEMA_NAME | VARCHAR2(128) | | For table and schema rules, the schema name |
| OBJECT_NAME | VARCHAR2(128) | | For table rules, the table name |
| INCLUDE_TAGGED_LCR | VARCHAR2(3) | | For global, schema or table rules, indicates whether to include tagged LCRs (YES) or not (NO) |
| SUBSETTING_OPERATION | VARCHAR2(6) | | For subset rules, the type of operation:<br>• INSERT<br>• UPDATE<br>• DELETE |
| DML_CONDITION | VARCHAR2(4000) | | For subset rules, the row subsetting condition |
| SOURCE_DATABASE | VARCHAR2(128) | | For global, schema or table rules, the name of the database where the LCRs originated |
| ORIGINAL_RULE_CONDITION | VARCHAR2(4000) | | For rules created by GoldenGate administrative APIs, the original rule condition when the rule was created |
| SAME_RULE_CONDITION | VARCHAR2(3) | | For rules created by GoldenGate administrative APIs, indicates whether the current rule condition is the same as the original rule condition (YES) or not (NO) |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database where the transactions originated |
| SOURCE_CONTAINER_NAME | VARCHAR2(128) | | The container name of the database where the transactions originated |

**See Also:**

"DBA_GOLDENGATE_RULES"

# ALL_HEAT_MAP_SEG_HISTOGRAM

ALL_HEAT_MAP_SEG_HISTOGRAM displays segment access information for all segments visible to the user.

### Related Views

- DBA_HEAT_MAP_SEG_HISTOGRAM displays segment access information for all segments.

- USER_HEAT_MAP_SEG_HISTOGRAM displays segment access information for segments owned by the user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Table owner |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the sub-object |
| TRACK_TIME | DATE | | System time when the segment access was tracked |
| SEGMENT_WRITE | VARCHAR2(3) | | Indicates whether the segment has write access (YES or NO) |
| FULL_SCAN | VARCHAR2(3) | | Indicates whether the segment has full table scan (YES or NO) |
| LOOKUP_SCAN | VARCHAR2(3) | | Indicates whether the segment has lookup scan (YES or NO) |

**See Also:**

- "DBA_HEAT_MAP_SEG_HISTOGRAM"
- "USER_HEAT_MAP_SEG_HISTOGRAM"

# ALL_HEAT_MAP_SEGMENT

ALL_HEAT_MAP_SEGMENT displays the latest segment access time for all segments visible to the user. The timestamps in the view are coarse with a granularity of a day reflecting the flush times of the heat map.

### Related Views

- DBA_HEAT_MAP_SEGMENT displays the latest segment access time for all segments.
- USER_HEAT_MAP_SEGMENT displays the latest segment access time for all segments owned by the user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Table owner |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the sub-object |
| SEGMENT_WRITE_TIME | DATE | | Latest timestamp on which the segment has write access |
| SEGMENT_READ_TIME | DATE | | Latest timestamp on which the segment has read access |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FULL_SCAN | DATE | | Latest timestamp on which the segment has full table scan |
| LOOKUP_SCAN | DATE | | Latest timestamp on which the segment has index scan |

**See Also:**

- "DBA_HEAT_MAP_SEGMENT"
- "USER_HEAT_MAP_SEGMENT"

# ALL_HISTOGRAMS

ALL_HISTOGRAMS is a synonym for ALL_TAB_HISTOGRAMS.

**See Also:**

"ALL_TAB_HISTOGRAMS"

# ALL_IDENTIFIERS

ALL_IDENTIFIERS displays information about the identifiers in the stored objects accessible to the current user.

### Related Views

- DBA_IDENTIFIERS displays information about the identifiers in all stored objects in the database.
- USER_IDENTIFIERS displays information about the identifiers in the stored objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the identifier |
| NAME | VARCHAR2(128) | | Name of the identifier |
| SIGNATURE | VARCHAR2(32) | | Signature of the identifier |
| TYPE | VARCHAR2(18) | | Type of the identifier |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object where the identifier action occurred |
| OBJECT_TYPE | VARCHAR2(13) | | Type of the object where the identifier action occurred |
| USAGE | VARCHAR2(11) | | Type of the identifier usage:<br>• DECLARATION<br>• DEFINITION |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • CALL<br>• REFERENCE<br>• ASSIGNMENT |
| USAGE_ID | NUMBER | | Unique key for the identifier usage within the object |
| LINE | NUMBER | | Line number of the identifier action |
| COL | NUMBER | | Column number of the identifier action |
| USAGE_CONTEXT_ID | NUMBER | | Context USAGE_ID of the identifier usage |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_IDENTIFIERS"
- "USER_IDENTIFIERS"

# ALL_IND_COLUMNS

ALL_IND_COLUMNS describes the columns of indexes on all tables accessible to the current user.

### Related Views

- DBA_IND_COLUMNS describes the columns of indexes on all tables in the database.
- USER_IND_COLUMNS describes the columns of indexes owned by the current user and columns of indexes on tables owned by the current user. This view does not display the INDEX_OWNER or TABLE_OWNER columns.

**Note:**

For join indexes, the TABLE_NAME and TABLE_OWNER columns in this view may not match the TABLE_NAME and TABLE_OWNER columns you find in the *_INDEXES (and other similar) data dictionary views.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_OWNER | VARCHAR2(128) | NOT NULL | Owner of the index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the index |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the table or cluster |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table or cluster |
| COLUMN_NAME | VARCHAR2(4000) | | Column name or attribute of the object type column<br><br>**Note:** If you create an index on a user-defined REF column, the system creates the index on the attributes that make up the REF column. Therefore, the column names displayed in this view are the attribute names, with the REF column name as a prefix, in the following form:<br>"REF_name"."attribute" |
| COLUMN_POSITION | NUMBER | NOT NULL | Position of the column or attribute within the index |
| COLUMN_LENGTH | NUMBER | NOT NULL | Indexed length of the column |
| CHAR_LENGTH | NUMBER | | Maximum codepoint length of the column |
| DESCEND | VARCHAR2(4) | | Indicates whether the column is sorted in descending order (DESC) or ascending order (ASC) |

**See Also:**

- "DBA_IND_COLUMNS"
- "USER_IND_COLUMNS"

# ALL_IND_EXPRESSIONS

ALL_IND_EXPRESSIONS describes the expressions of function-based indexes on tables accessible to the current user.

### Related Views

- DBA_IND_EXPRESSIONS describes the expressions of all function-based indexes in the database.
- USER_IND_EXPRESSIONS describes the expressions of function-based indexes on tables owned by the current user. This view does not display the INDEX_OWNER or TABLE_OWNER columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_OWNER | VARCHAR2(128) | NOT NULL | Owner of the index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the index |
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the table or cluster |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table or cluster |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COLUMN_EXPRESSION | LONG | | Function-based index expression defining the column |
| COLUMN_POSITION | NUMBER | NOT NULL | Position of the column or attribute within the index |

**See Also:**

- "DBA_IND_EXPRESSIONS"
- "USER_IND_EXPRESSIONS"

# ALL_IND_PARTITIONS

ALL_IND_PARTITIONS displays, for each index partition accessible to the current user, the partition-level partitioning information, the storage parameters for the partition, and various partition statistics generated by the DBMS_STATS package.

### Related Views

- DBA_IND_PARTITIONS describes all index partitions in the database.
- USER_IND_PARTITIONS describes the index partitions owned by the current user. This view does not display the INDEX_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_OWNER | VARCHAR2(128) | | Owner of the index |
| INDEX_NAME | VARCHAR2(128) | | Name of the index |
| COMPOSITE | VARCHAR2(3) | | Indicates whether the partition belongs to a local index on a composite-partitioned table (YES) or not (NO) |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_COUNT | NUMBER | | If a local index on a composite-partitioned table, the number of subpartitions in the partition |
| HIGH_VALUE | LONG | | Partition bound value expression |
| HIGH_VALUE_LENGTH | NUMBER | | Length of the partition bound value expression |
| PARTITION_POSITION | NUMBER | | Position of the partition within the index |
| STATUS | VARCHAR2(8) | | Indicates whether the index partition is usable (USABLE) or not (UNUSABLE) |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the partition |
| PCT_FREE | NUMBER | | Minimum percentage of free space in a block |
| INI_TRANS | NUMBER | | Initial number of transactions |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MAX_TRANS | NUMBER | | Maximum number of transactions |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent in bytes |
| NEXT_EXTENT | NUMBER | | Size of secondary extents in bytes |
| MIN_EXTENT | NUMBER | | Minimum number of extents allowed in the segment |
| MAX_EXTENT | NUMBER | | Maximum number of extents allowed in the segment |
| MAX_SIZE | NUMBER | | Maximum number of blocks allowed in the segment |
| PCT_INCREASE | NUMBER | | Percentage increase in extent size |
| FREELISTS | NUMBER | | Number of process freelists allocated in this segment |
| FREELIST_GROUPS | NUMBER | | Number of process freelist groups allocated in this segment |
| LOGGING | VARCHAR2(7) | | Indicates whether or not changes to the index are logged:<br>• NONE - Not specified<br>**See Also:** the *_IND_SUBPARTITIONS view<br>• YES<br>• NO |
| COMPRESSION | VARCHAR2(13) | | Indicates whether index compression is enabled or disabled for a partitioned index; NULL for a nonpartitioned index:<br>• NONE - Not specified<br>**See Also:** the *_IND_SUBPARTITIONS view<br>• ENABLED<br>• DISABLED |
| BLEVEL | NUMBER | | B*-Tree level (depth of the index from its root block to its leaf blocks). A depth of 0 indicates that the root block and leaf block are the same. |
| LEAF_BLOCKS | NUMBER | | Number of leaf blocks in the index partition |
| DISTINCT_KEYS | NUMBER | | Number of distinct keys in the index partition |
| AVG_LEAF_BLOCKS_PER_KEY | NUMBER | | Average number of leaf blocks in which each distinct value in the index appears, rounded to the nearest integer. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is always 1. |
| AVG_DATA_BLOCKS_PER_KEY | NUMBER | | Average number of data blocks in the table that are pointed to by a distinct value in the index rounded to the nearest integer. This statistic is the average number of data blocks that contain rows that contain a given value for the indexed columns. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLUSTERING_FACTOR | NUMBER | | Indicates the amount of order of the rows in the table based on the values of the index.<br>• If the value is near the number of blocks, then the table is very well ordered. In this case, the index entries in a single leaf block tend to point to rows in the same data blocks.<br>• If the value is near the number of rows, then the table is very randomly ordered. In this case, it is unlikely that index entries in the same leaf block point to rows in the same data blocks. |
| NUM_ROWS | NUMBER | | Number of rows returned |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this partition |
| LAST_ANALYZED | DATE | | Date on which this partition was most recently analyzed |
| BUFFER_POOL | VARCHAR2(7) | | Actual buffer pool for the partition:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for partition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for partition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| PCT_DIRECT_ACCESS | NUMBER | | If a secondary index on index-organized table, the percentage of rows with VALID guess |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the partition as a whole (YES) or they were not gathered (NO). Note that for partitioned indexes, partition statistics are estimated from subpartition level statistics when set to NO. |
| DOMIDX_OPSTATUS | VARCHAR2(6) | | Status of the operation on a domain index:<br>• NULL - Index is not a domain index<br>• VALID - Operation performed without errors<br>• FAILED - Operation failed with an error |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETERS | VARCHAR2(1000) | | For a domain index, the parameter string |
| INTERVAL | VARCHAR2(3) | | Indicates whether the partition is in the interval section of an interval partitioned table (YES) or whether the partition is in the range section (NO) |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the index partition segment has been created (YES) or not (NO); N/A indicates that this index is subpartitioned and no segment exists at the partition level |
| ORPHANED_ENTRIES | VARCHAR2(3) | | Indicates whether a global index partition contains stale entries because of deferred index maintenance during DROP/TRUNCATE PARTITION, or MODIFY PARTITION INDEXING OFF operations.<br>Possible values:<br>• YES - the index partition contains orphaned entries<br>• NO - the index partition does not contain orphaned entries |

**See Also:**

- "DBA_IND_PARTITIONS"
- "USER_IND_PARTITIONS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_IND_PENDING_STATS

ALL_IND_PENDING_STATS describes the pending statistics for tables, partitions, and subpartitions accessible to the current user collected using the DBMS_STATS package.

### Related Views

- DBA_IND_PENDING_STATS describes pending statistics for all tables, partitions, and subpartitions in the database.
- USER_IND_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Name of the index owner |
| INDEX_NAME | VARCHAR2(128) | | Index name |
| TABLE_OWNER | VARCHAR2(128) | | Table owner name |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| BLEVEL | NUMBER | | Number of levels in the index |
| LEAF_BLOCKS | NUMBER | | Number of leaf blocks in the index |
| DISTINCT_KEYS | NUMBER | | Number of distinct keys in the index |
| AVG_LEAF_BLOCKS_PER_KEY | NUMBER | | Average number of leaf blocks per key |
| AVG_DATA_BLOCKS_PER_KEY | NUMBER | | Average number of data blocks per key |
| CLUSTERING_FACTOR | NUMBER | | Clustering factor |
| NUM_ROWS | NUMBER | | Number of rows in the index |
| SAMPLE_SIZE | NUMBER | | Sample size |
| LAST_ANALYZED | DATE | | Time of the last analysis |

**See Also:**

- "DBA_IND_PENDING_STATS"
- "USER_IND_PENDING_STATS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_IND_STATISTICS

ALL_IND_STATISTICS displays optimizer statistics for the indexes on the tables accessible to the current user collected using the DBMS_STATS package.

### Related Views

- DBA_IND_STATISTICS displays optimizer statistics for all indexes in the database.
- USER_IND_STATISTICS displays optimizer statistics for the indexes on the tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the index |
| INDEX_NAME | VARCHAR2(128) | | Name of the index |
| TABLE_OWNER | VARCHAR2(128) | | Owner of the indexed object |
| TABLE_NAME | VARCHAR2(128) | | Name of the indexed object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| PARTITION_POSITION | NUMBER | | Position of the partition within the index |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| SUBPARTITION_POSITION | NUMBER | | Position of the subpartition within the partition |
| OBJECT_TYPE | VARCHAR2(12) | | Type of the object:<br>• INDEX<br>• PARTITION<br>• SUBPARTITION |
| BLEVEL | NUMBER | | B-Tree level |
| LEAF_BLOCKS | NUMBER | | Number of leaf blocks in the index |
| DISTINCT_KEYS | NUMBER | | Number of distinct keys in the index |
| AVG_LEAF_BLOCKS_PER_KEY | NUMBER | | Average number of leaf blocks per key |
| AVG_DATA_BLOCKS_PER_KEY | NUMBER | | Average number of data blocks per key |
| CLUSTERING_FACTOR | NUMBER | | Indicates the amount of order of the rows in the table based on the values of the index.<br>• If the value is near the number of blocks, then the table is very well ordered. In this case, the index entries in a single leaf block tend to point to rows in the same data blocks.<br>• If the value is near the number of rows, then the table is very randomly ordered. In this case, it is unlikely that index entries in the same leaf block point to rows in the same data blocks. |
| NUM_ROWS | NUMBER | | Number of rows in the index |
| AVG_CACHED_BLOCKS | NUMBER | | Average number of blocks in the buffer cache |
| AVG_CACHE_HIT_RATIO | NUMBER | | Average cache hit ratio for the object |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing the index |
| LAST_ANALYZED | DATE | | Date of the most recent time the index was analyzed |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered on the index (YES) or they were not gathered (NO). Note that for partitioned indexes, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATTYPE_LOCKED | VARCHAR2(5) | | Type of statistics lock |
| STALE_STATS | VARCHAR2(3) | | Whether statistics for the object are stale or not |
| SCOPE | VARCHAR2(7) | | The value is SHARED for statistics gathered on any table other than global temporary tables.<br>For a global tempoary table, the possible values are:<br>• SESSION - Indicates that the statistics are session-specific<br>• SHARED - Indicates that the statistics are shared across all sessions<br>See *Oracle Database PL/SQL Packages and Types Reference* for information about using the GLOBAL_TEMP_TABLE_STATS preference of the DBMS_STATS package to control whether to gather session or shared statistics for global temporary tables. |

**See Also:**

- "DBA_IND_STATISTICS"
- "USER_IND_STATISTICS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_IND_SUBPARTITIONS

ALL_IND_SUBPARTITIONS displays, for each index subpartition accessible to the current user, the subpartition-level partitioning information, the storage parameters for the subpartition, and various subpartition statistics generated by the DBMS_STATS package.

### Related Views

- DBA_IND_SUBPARTITIONS describes all index subpartitions in the database.
- USER_IND_SUBPARTITIONS describes the index subpartitions owned by the current user. This view does not display the INDEX_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_OWNER | VARCHAR2(128) | NOT NULL | Owner of the index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the index |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| HIGH_VALUE | LONG | | Subpartition bound value expression |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HIGH_VALUE_LENGTH | NUMBER | NOT NULL | Length of the subpartition bound value expression |
| SUBPARTITION_POSITION | NUMBER | | Position of a subpartition within a partition |
| STATUS | VARCHAR2(8) | | Indicates whether the index partition is usable (USABLE) or not (UNUSABLE) |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace containing the partition |
| PCT_FREE | NUMBER | NOT NULL | Minimum percentage of free space in a block |
| INI_TRANS | NUMBER | NOT NULL | Initial number of transactions |
| MAX_TRANS | NUMBER | NOT NULL | Maximum number of transactions |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent in bytes |
| NEXT_EXTENT | NUMBER | | Size of secondary extents in bytes |
| MIN_EXTENT | NUMBER | | Minimum number of extents allowed in the segment |
| MAX_EXTENT | NUMBER | | Maximum number of extents allowed in the segment |
| MAX_SIZE | NUMBER | | Maximum number of blocks allowed in the segment |
| PCT_INCREASE | NUMBER | NOT NULL | Percentage increase in extent size |
| FREELISTS | NUMBER | | Number of process freelists allocated in this segment |
| FREELIST_GROUPS | NUMBER | | Number of process freelist groups allocated in this segment |
| LOGGING | VARCHAR2(3) | | Indicates whether or not changes to the index are logged:<br>• YES<br>• NO |
| COMPRESSION | VARCHAR2(13) | | Indicates whether this subpartition is compressed (ENABLED) or not (DISABLED) |
| BLEVEL | NUMBER | | B-Tree level (depth of the index from its root block to its leaf blocks). A depth of 0 indicates that the root block and leaf block are the same. |
| LEAF_BLOCKS | NUMBER | | Number of leaf blocks in the index |
| DISTINCT_KEYS | NUMBER | | Number of distinct keys in the index partition |
| AVG_LEAF_BLOCKS_PER_KEY | NUMBER | | Average number of leaf blocks in which each distinct value in the index appears, rounded to the nearest integer. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is always 1. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AVG_DATA_BLOCKS_PER_KEY | NUMBER | | Average number of data blocks in the table that are pointed to by a distinct value in the index rounded to the nearest integer. This statistic is the average number of data blocks that contain rows that contain a given value for the indexed columns. |
| CLUSTERING_FACTOR | NUMBER | | Indicates the amount of order of the rows in the table based on the values of the index.<br>• If the value is near the number of blocks, then the table is very well ordered. In this case, the index entries in a single leaf block tend to point to rows in the same data blocks.<br>• If the value is near the number of rows, then the table is very randomly ordered. In this case, it is unlikely that index entries in the same leaf block point to rows in the same data blocks. |
| NUM_ROWS | NUMBER | | Number of rows in this index subpartition |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this subpartition |
| LAST_ANALYZED | DATE | | Date on which this partition was most recently analyzed |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool for the subpartition:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for subpartition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for subpartition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered on a subpartition (YES) or they were not gathered (NO) |
| INTERVAL | VARCHAR2(3) | | Indicates whether the partition is in the interval section of an interval partitioned table (YES) or whether the partition is in the range section (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the index subpartition segment has been created (YES) or not (NO); N/A indicates that this index is not subpartitioned |

**See Also:**

- "DBA_IND_SUBPARTITIONS"
- "USER_IND_SUBPARTITIONS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_INDEXES

ALL_INDEXES describes the indexes on the tables accessible to the current user. To gather statistics for this view and the related views DBA_INDEXES and USER_INDEXES, use the DBMS_STATS package.

### Related Views

- DBA_INDEXES describes all indexes in the database.
- USER_INDEXES describes the indexes owned by the current user. This view does not display the OWNER column.

**Note:**

Column names followed by an asterisk are populated only if you collect statistics on the index using the DBMS_STATS package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the index |
| INDEX_TYPE | VARCHAR2(27) | | Type of the index:<br>• LOB<br>• NORMAL<br>• NORMAL/REV<br>• BITMAP<br>• FUNCTION-BASED NORMAL<br>• FUNCTION-BASED NORMAL/REV<br>• FUNCTION-BASED BITMAP<br>• FUNCTION-BASED DOMAIN<br>• CLUSTER<br>• IOT - TOP<br>• DOMAIN |
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the indexed object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the indexed object |
| TABLE_TYPE | CHAR(11) | | Type of the indexed object:<br>• NEXT OBJECT<br>• INDEX<br>• TABLE<br>• CLUSTER<br>• VIEW<br>• SYNONYM<br>• SEQUENCE |
| UNIQUENESS | VARCHAR2(9) | | Indicates whether the index is unique (UNIQUE) or nonunique (NONUNIQUE) |
| COMPRESSION | VARCHAR2(13) | | The type of compression being used for the index:<br>• ADVANCED LOW - Advanced low compression<br>• DISABLED - No compression is present<br>• ENABLED - Prefix compression |
| PREFIX_LENGTH | NUMBER | | Number of columns in the prefix of the compression key |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the index |
| INI_TRANS | NUMBER | | Initial number of transactions |
| MAX_TRANS | NUMBER | | Maximum number of transactions |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent |
| NEXT_EXTENT | NUMBER | | Size of secondary extents |
| MIN_EXTENTS | NUMBER | | Minimum number of extents allowed in the segment |
| MAX_EXTENTS | NUMBER | | Maximum number of extents allowed in the segment |
| PCT_INCREASE | NUMBER | | Percentage increase in extent size |
| PCT_THRESHOLD | NUMBER | | Threshold percentage of block space allowed per index entry |
| INCLUDE_COLUMN | NUMBER | | Column ID of the last column to be included in index-organized table primary key (non-overflow) index. This column maps to the COLUMN_ID column of the *_TAB_COLUMNS view. |
| FREELISTS | NUMBER | | Number of process freelists allocated to this segment |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated to this segment |
| PCT_FREE | NUMBER | | Minimum percentage of free space in a block |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOGGING | VARCHAR2(3) | | Indicates whether or not changes to the index are logged:<br>• YES<br>• NO |
| BLEVEL* | NUMBER | | B*-Tree level (depth of the index from its root block to its leaf blocks). A depth of 0 indicates that the root block and leaf block are the same. |
| LEAF_BLOCKS* | NUMBER | | Number of leaf blocks in the index |
| DISTINCT_KEYS* | NUMBER | | Number of distinct indexed values. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the table (*_TABLES.NUM_ROWS) |
| AVG_LEAF_BLOCKS_PER_KEY* | NUMBER | | Average number of leaf blocks in which each distinct value in the index appears, rounded to the nearest integer. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is always 1. |
| AVG_DATA_BLOCKS_PER_KEY* | NUMBER | | Average number of data blocks in the table that are pointed to by a distinct value in the index rounded to the nearest integer. This statistic is the average number of data blocks that contain rows that contain a given value for the indexed columns. |
| CLUSTERING_FACTOR* | NUMBER | | Indicates the amount of order of the rows in the table based on the values of the index.<br>• If the value is near the number of blocks, then the table is very well ordered. In this case, the index entries in a single leaf block tend to point to rows in the same data blocks.<br>• If the value is near the number of rows, then the table is very randomly ordered. In this case, it is unlikely that index entries in the same leaf block point to rows in the same data blocks.<br>For bitmap indexes, this column is not applicable. |
| STATUS | VARCHAR2(8) | | Indicates whether a nonpartitioned index is VALID or UNUSABLE |
| NUM_ROWS | NUMBER | | Number of rows in the index.<br>For bitmap indexes, this column is the number of distinct keys, so its value is the same as the DISTINCT_KEYS column. |
| SAMPLE_SIZE | NUMBER | | Size of the sample used to analyze the index |
| LAST_ANALYZED | DATE | | Date on which this index was most recently analyzed |
| DEGREE | VARCHAR2(40) | | Number of threads per instance for scanning the index, or DEFAULT |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INSTANCES | VARCHAR2(40) | | Number of instances across which the indexes to be scanned, or DEFAULT |
| PARTITIONED | VARCHAR2(3) | | Indicates whether the index is partitioned (YES) or not (NO) |
| TEMPORARY | VARCHAR2(1) | | Indicates whether the index is on a temporary table (Y) or not (N) |
| GENERATED | VARCHAR2(1) | | Indicates whether the name of the index is system-generated (Y) or not (N) |
| SECONDARY | VARCHAR2(1) | | Indicates whether the index is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N) |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool to be used for index blocks:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for index blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for index blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| DURATION | VARCHAR2(15) | | Indicates the duration of a temporary table:<br>• SYS$SESSION - Rows are preserved for the duration of the session<br>• SYS$TRANSACTION - Rows are deleted after COMMIT<br>NNUL - Permanent table |
| PCT_DIRECT_ACCESS | NUMBER | | For a secondary index on an index-organized table, the percentage of rows with VALID guess |
| ITYP_OWNER | VARCHAR2(128) | | For a domain index, the owner of the indextype |
| ITYP_NAME | VARCHAR2(128) | | For a domain index, the name of the indextype |
| PARAMETERS | VARCHAR2(1000) | | For a domain index, the parameter string |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the index (YES) or they were not gathered (NO). Note that for partitioned indexes, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| DOMIDX_STATUS | VARCHAR2(12) | | Status of a domain index:<br>• NULL - Index is not a domain index<br>• VALID - Index is a valid domain index<br>• IDXTYP_INVLD - Indextype of the domain index is invalid |
| DOMIDX_OPSTATUS | VARCHAR2(6) | | Status of the operation on a domain index:<br>• NULL - Index is not a domain index<br>• VALID - Operation performed without errors<br>• FAILED - Operation failed with an error |
| FUNCIDX_STATUS | VARCHAR2(8) | | Status of a function-based index:<br>• NULL - Index is not a function-based index<br>• ENABLED - Function-based index is enabled<br>• DISABLED - Function-based index is disabled |
| JOIN_INDEX | VARCHAR2(3) | | Indicates whether the index is a join index (YES) or not (NO) |
| IOT_REDUNDANT_PKEY_ELIM | VARCHAR2(3) | | Indicates whether redundant primary key columns are eliminated from secondary indexes on index-organized tables (YES) or not (NO) |
| DROPPED | VARCHAR2(3) | | Indicates whether the index has been dropped and is in the recycle bin (YES) or not (NO); NULL for partitioned tables |
| VISIBILITY | VARCHAR2(9) | | Indicates whether the index is VISIBLE or INVISIBLE to the optimizer |
| DOMIDX_MANAGEMENT | VARCHAR2(14) | | If this is a domain index, indicates whether the domain index is system-managed (SYSTEM_MANAGED) or user-managed (USER_MANAGED) |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the index segment has been created (YES) or not (NO) |
| ORPHANED_ENTRIES | VARCHAR2(3) | | Indicates whether a global index contains stale entries because of deferred index maintenance during DROP/TRUNCATE PARTITION, or MODIFY PARTITION INDEXING OFF operations.<br>Possible values:<br>• YES - The index contains orphaned entries<br>• NO - The index does not contain orphaned entries |
| INDEXING | VARCHAR2(7) | | Indicates whether a global index is decoupled from the underlying table. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | Possible values:<br>• PARTIAL - The index is partial, that is, it will follow the table's indexing property.<br>• FULL - The index will include all partitions of the table. |

**See Also:**

- "DBA_INDEXES"
- "USER_INDEXES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_INDEXTYPE_ARRAYTYPES

ALL_INDEXTYPE_ARRAYTYPES displays information about the array types specified by the indextypes accessible to the current user.

### Related Views

- DBA_INDEXTYPE_ARRAYTYPES displays information about the array types specified by all indextypes in the database.
- USER_INDEXTYPE_ARRAYTYPES displays information about the array types specified by the indextypes owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the indextype |
| INDEXTYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the indextype |
| BASE_TYPE_SCHEMA | VARCHAR2(128) | | Name of the base type schema |
| BASE_TYPE_NAME | VARCHAR2(128) | | Name of the base type name |
| BASE_TYPE | VARCHAR2(30) | | Datatype of the base type |
| ARRAY_TYPE_SCHEMA | VARCHAR2(128) | NOT NULL | Name of the array type schema |
| ARRAY_TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the array type name |

**See Also:**

- "DBA_INDEXTYPE_ARRAYTYPES"
- "USER_INDEXTYPE_ARRAYTYPES"

# ALL_INDEXTYPE_COMMENTS

ALL_INDEXTYPE_COMMENTS displays comments for the user-defined indextypes accessible to the current user.

### Related Views

- DBA_INDEXTYPE_COMMENTS displays comments for all user-defined indextypes in the database.
- USER_INDEXTYPE_COMMENTS displays comments for the user-defined indextypes owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the user-defined indextype |
| INDEXTYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the user-defined indextype |
| COMMENTS | VARCHAR2(4000) | | Comment for the user-defined indextype |

**See Also:**

- "DBA_INDEXTYPE_COMMENTS"
- "USER_INDEXTYPE_COMMENTS"

# ALL_INDEXTYPE_OPERATORS

ALL_INDEXTYPE_OPERATORS lists all operators supported by indextypes accessible to the current user.

### Related Views

- DBA_INDEXTYPE_OPERATORS lists all operators supported by indextypes in the database.
- USER_INDEXTYPE_OPERATORS lists all operators supported by indextypes owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the indextype |
| INDEXTYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the indextype |
| OPERATOR_SCHEMA | VARCHAR2(128) | NOT NULL | Name of the operator schema |
| OPERATOR_NAME | VARCHAR2(128) | NOT NULL | Name of the operator for which the indextype is defined |
| BINDING# | NUMBER | NOT NULL | Binding number associated with the operator |

**See Also:**

- "DBA_INDEXTYPE_OPERATORS"
- "USER_INDEXTYPE_OPERATORS"

# ALL_INDEXTYPES

ALL_INDEXTYPES displays information about the indextypes accessible to the current user.

### Related Views

- DBA_INDEXTYPES displays information about all indextypes in the database.
- USER_INDEXTYPES displays information about the indextypes owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the indextype |
| INDEXTYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the indextype |
| IMPLEMENTATION_SCHEMA | VARCHAR2(128) | NOT NULL | Name of the schema for the indextype implementation (that is, containing the indextype operators) |
| IMPLEMENTATION_NAME | VARCHAR2(128) | NOT NULL | Name of the indextype implementation type |
| INTERFACE_VERSION | NUMBER | | Version of the indextype interface |
| IMPLEMENTATION_VERSION | NUMBER | NOT NULL | Version of the indextype implementation |
| NUMBER_OF_OPERATORS | NUMBER | | Number of operators associated with the indextype |
| PARTITIONING | VARCHAR2(10) | | Kinds of local partitioning supported by the indextype:<br>• NONE - Indextype does not support local domain indexes<br>• RANGE - Indextype can support range partitioned local user managed domain indexes<br>• LOCAL - Indextype can support local system managed domain indexes (range, list, hash, or interval) |
| ARRAY_DML | VARCHAR2(3) | | Indicates whether the indextype supports array DML (YES) or not (NO) |
| MAINTENANCE_TYPE | VARCHAR2(14) | | Indicates whether the indextype is system-managed (SYSTEM_MANAGED) or user-managed (USER_MANAGED) |

**See Also:**

- "DBA_INDEXTYPES"
- "USER_INDEXTYPES"

# ALL_INTERNAL_TRIGGERS

ALL_INTERNAL_TRIGGERS describes internal triggers on tables accessible to the current user. Internal triggers are internal pieces of code executed when a particular flag is set for a table. This view does not display the OWNER_NAME column.

### Related Views

- DBA_INTERNAL_TRIGGERS describes internal triggers on all tables in the database.
- USER_INTERNAL_TRIGGERS describes all internal triggers on tables owned by the current user. This view does not display the OWNER_NAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | | Name of the table on which the trigger is defined |
| INTERNAL_TRIGGER_TYPE | VARCHAR2(19) | | Indicates the type of internal trigger on the table |

**See Also:**

- "DBA_INTERNAL_TRIGGERS"
- "USER_INTERNAL_TRIGGERS"

# ALL_JAVA_ARGUMENTS

ALL_JAVA_ARGUMENTS displays argument information about the stored Java classes accessible to the current user.

### Related Views

- DBA_JAVA_ARGUMENTS displays argument information about all stored Java classes in the database.
- USER_JAVA_ARGUMENTS displays argument information about the stored Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |
| METHOD_INDEX | NUMBER | | Index of the hosting method of the argument |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| METHOD_NAME | VARCHAR2(4000) | | Name of the hosting method of the argument |
| ARGUMENT_POSITION | NUMBER | | Position of the argument, starting from 0 |
| ARRAY_DEPTH | NUMBER | | Array depth of the type of the argument |
| BASE_TYPE | VARCHAR2(7) | | Base type of the type of the argument:<br>• int<br>• long<br>• float<br>• double<br>• boolean<br>• byte<br>• char<br>• short<br>• class |
| ARGUMENT_CLASS | VARCHAR2(4000) | | Actual class name of the argument if the base type is class |

**See Also:**

- "DBA_JAVA_ARGUMENTS"
- "USER_JAVA_ARGUMENTS"

# ALL_JAVA_CLASSES

ALL_JAVA_CLASSES displays class level information about the stored Java classes accessible to the current user.

### Related Views

- DBA_JAVA_CLASSES displays class level information about all stored Java classes in the database.
- USER_JAVA_CLASSES displays class level information about the stored Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |
| MAJOR | NUMBER | | Major version number of the Java class, as defined in the JVM specification |
| MINOR | NUMBER | | Minor version number of the Java class, as defined in the JVM specification |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| KIND | VARCHAR2(50) | | Indicates whether the stored object is a Java class (CLASS) or a Java interface (INTERFACE) |
| ACCESSIBILITY | VARCHAR2(9) | | Accessibility of the Java class |
| IS_INNER | VARCHAR2(3) | | Indicates whether this Java class is an inner class (YES) or not (NO) |
| IS_ABSTRACT | VARCHAR2(3) | | Indicates whether this Java class is an abstract class (YES) or not (NO) |
| IS_FINAL | VARCHAR2(3) | | Indicates whether this Java class is a final class (YES) or not (NO) |
| IS_STATIC | VARCHAR2(3) | | Indicates whether this is a static inner class (YES) or not (NO) |
| IS_STRICTFP | VARCHAR2(3) | | Indicates whether the class is declared strict floating point for portability (YES) or not (NO) |
| IS_SYNTHETIC | VARCHAR2(3) | | Indicates whether this is an inner class generated by the compiler (YES) or not (NO) |
| IS_DEBUG | VARCHAR2(3) | | Indicates whether this Java class contains debug information (YES) or not (NO) |
| SOURCE | VARCHAR2(4000) | | Source designation of the Java class |
| SUPER | VARCHAR2(4000) | | Super class of this Java class |
| OUTER | VARCHAR2(4000) | | Outer class of this Java class if this Java class is an inner class |

**See Also:**

- "DBA_JAVA_CLASSES"
- "USER_JAVA_CLASSES"

# ALL_JAVA_COMPILER_OPTIONS

ALL_JAVA_COMPILER_OPTIONS displays information about the native compiler options accessible to the current user.

### Related Views

- DBA_JAVA_COMPILER_OPTIONS displays information about all native compiler options in the database.
- USER_JAVA_COMPILER_OPTIONS displays information about the native compiler options owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the native compiler option |
| OPTION_NAME | VARCHAR2(64) | | Name of the native compiler option |
| VALUE | VARCHAR2(4000) | | Value of the native compiler option |

**See Also:**

- "DBA_JAVA_COMPILER_OPTIONS"
- "USER_JAVA_COMPILER_OPTIONS"

# ALL_JAVA_DERIVATIONS

ALL_JAVA_DERIVATIONS displays mapping information about Java source objects and their derived Java class objects and Java resource objects for the Java classes accessible to the current user.

### Related Views

- DBA_JAVA_DERIVATIONS displays mapping information about Java source objects and their derived Java class objects and Java resource objects for all Java classes in the database.
- USER_JAVA_DERIVATIONS displays mapping information about Java source objects and their derived Java class objects and Java resource objects for the Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java source object |
| SOURCE_NAME | VARCHAR2(4000) | | Name of the Java source object |
| CLASS_INDEX | NUMBER | | Index of the derived Java class object |
| CLASS_NAME | VARCHAR2(4000) | | Name of the derived Java class object |

**See Also:**

- "DBA_JAVA_DERIVATIONS"
- "USER_JAVA_DERIVATIONS"

# ALL_JAVA_FIELDS

ALL_JAVA_FIELDS displays field information about the stored Java classes accessible to the current user.

### Related Views

- DBA_JAVA_FIELDS displays field information about all stored Java classes in the database.
- USER_JAVA_FIELDS displays field information about the stored Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |
| FIELD_INDEX | NUMBER | | Index of the field |
| FIELD_NAME | VARCHAR2(4000) | | Name of the field identified by the FIELD_INDEX column |
| ACCESSIBILITY | VARCHAR2(9) | | Accessibility of the field:<br>• PUBLIC<br>• PRIVATE<br>• PROTECTED |
| IS_STATIC | VARCHAR2(3) | | Indicates whether the field is a static field (YES) or not (NO) |
| IS_FINAL | VARCHAR2(3) | | Indicates whether the field is a final field (YES) or not (NO) |
| IS_VOLATILE | VARCHAR2(3) | | Indicates whether the field is volatile (YES) or not (NO) |
| IS_TRANSIENT | VARCHAR2(3) | | Indicates whether the field is transient (YES) or not (NO) |
| IS_SYNTHETIC | VARCHAR2(3) | | Indicates whether the field is generated by the compiler (YES) or not (NO) |
| IS_ENUM | VARCHAR2(3) | | Indicates whether the field is a member of an enum (YES) or not (NO) |
| ARRAY_DEPTH | NUMBER | | Array depth of the type of the field |
| BASE_TYPE | VARCHAR2(7) | | Base type of the type of the field:<br>• int<br>• long<br>• float<br>• double<br>• boolean<br>• byte<br>• char<br>• short<br>• class |
| FIELD_CLASS | VARCHAR2(4000) | | Actual class name of the base object if the base type is class |

**See Also:**

- "DBA_JAVA_FIELDS"
- "USER_JAVA_FIELDS"

# ALL_JAVA_IMPLEMENTS

ALL_JAVA_IMPLEMENTS describes interfaces implemented by the stored Java classes accessible to the current user.

### Related Views

- DBA_JAVA_IMPLEMENTS describes interfaces implemented by all stored Java classes in the database.
- USER_JAVA_IMPLEMENTS describes interfaces implemented by the stored Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |
| INTERFACE_INDEX | NUMBER | | Index of the interfaces implemented by the Java class |
| INTERFACE_NAME | VARCHAR2(4000) | | Name of the interface identified by the INTERFACE_INDEX column |

**See Also:**

- "DBA_JAVA_IMPLEMENTS"
- "USER_JAVA_IMPLEMENTS"

# ALL_JAVA_INNERS

ALL_JAVA_INNERS displays information about inner classes referred to by the stored Java classes accessible to the current user.

### Related Views

- DBA_JAVA_INNERS displays information about inner classes referred to by all stored Java classes in the database.
- USER_JAVA_INNERS displays information about inner classes referred to by the stored Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |
| INNER_INDEX | NUMBER | | Index of the referred inner class |
| SIMPLE_NAME | VARCHAR2(4000) | | Simple name of the referred inner class |
| FULL_NAME | VARCHAR2(4000) | | Full name of the referred inner class |
| ACCESSIBILITY | VARCHAR2(9) | | Accessibility of the referred inner class:<br>• PUBLIC<br>• PRIVATE<br>• PROTECTED |
| IS_STATIC | VARCHAR2(3) | | Indicates whether the referred inner class is declared static in the source file (YES) or not (NO) |
| IS_FINAL | VARCHAR2(3) | | Indicates whether the referred inner class is declared final in the source file (YES) or not (NO) |
| IS_ABSTRACT | VARCHAR2(3) | | Indicates whether the referred inner class is declared abstract in the source file (YES) or not (NO) |
| IS_INTERFACE | VARCHAR2(3) | | Indicates whether the referred inner class is declared interface in the source file (YES) or not (NO) |
| IS_STRICTFP | VARCHAR2(3) | | Indicates whether the inner class is declared strictfp (YES) or not (NO) |
| IS_SYNTHETIC | VARCHAR2(3) | | Indicates whether the inner class is generated by the compiler (YES) or not (NO) |
| IS_ENUM | VARCHAR2(3) | | Indicates whether the inner class is an enum (YES) or not (NO) |
| IS_ANNOTATION | VARCHAR2(3) | | Indicates whether the inner class is an annotation (YES) or not (NO) |
| KIND | VARCHAR2(50) | | Type (class, interface, enum or annotation) of the inner class |
| ALL_QUALIFIERS | VARCHAR2(101) | | Concatenation of accessibility, type and other attributes of the inner class |

**See Also:**

- "DBA_JAVA_INNERS"
- "USER_JAVA_INNERS"

# ALL_JAVA_LAYOUTS

ALL_JAVA_LAYOUTS displays class layout information about the stored Java classes accessible to the current user.

### Related Views

- DBA_JAVA_LAYOUTS displays class layout information about all stored Java classes in the database.
- USER_JAVA_LAYOUTS displays class layout information about the stored Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |
| INTERFACES | NUMBER | | Number of interfaces that this Java class implements |
| INNER_CLASSES | NUMBER | | Number of inner classes that this Java class contains |
| FIELDS | NUMBER | | Number of locally declared fields that this Java class contains |
| STATIC_FIELDS | NUMBER | | Number of locally declared static fields that this Java class contains |
| METHODS | NUMBER | | Number of locally declared methods that this Java class contains |
| STATIC_METHODS | NUMBER | | Number of locally declared static methods that this Java class contains |
| NATIVE_METHODS | NUMBER | | Number of locally declared native methods that this Java class contains |

**See Also:**

- "DBA_JAVA_LAYOUTS"
- "USER_JAVA_LAYOUTS"

# ALL_JAVA_METHODS

ALL_JAVA_METHODS displays method information about the stored Java classes accessible to the current user.

### Related Views

- DBA_JAVA_METHODS displays method information about all stored Java classes in the database.

- USER_JAVA_METHODS displays method information about the stored Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |
| METHOD_INDEX | NUMBER | | Index of the method |
| METHOD_NAME | VARCHAR2(4000) | | Name of the method identified by the METHOD_INDEX column |
| ACCESSIBILITY | VARCHAR2(9) | | Accessibility of the method:<br>• PUBLIC<br>• PRIVATE<br>• PROTECTED |
| IS_STATIC | VARCHAR2(3) | | Indicates whether the method is a static method (YES) or not (NO) |
| IS_FINAL | VARCHAR2(3) | | Indicates whether the method is a final method (YES) or not (NO) |
| IS_SYNCHRONIZED | VARCHAR2(3) | | Indicates whether the method is a synchronized method (YES) or not (NO) |
| HAS_VARARGS | VARCHAR2(3) | | Indicates whether the method has a variable number of arguments (YES) or not (NO) |
| IS_NATIVE | VARCHAR2(3) | | Indicates whether the method is a native method (YES) or not (NO) |
| IS_ABSTRACT | VARCHAR2(3) | | Indicates whether the method is an abstract method (YES) or not (NO) |
| IS_STRICTFP | VARCHAR2(3) | | Indicates whether the method is a strict method (YES) or not (NO) |
| IS_SYNTHETIC | VARCHAR2(3) | | Indicates whether the method is generated by the compiler (YES) or not (NO) |
| ARGUMENTS | NUMBER | | Number of arguments of the method |
| THROWS | NUMBER | | Number of exceptions thrown by the method |
| ARRAY_DEPTH | NUMBER | | Array depth of the return type of the method |
| BASE_TYPE | VARCHAR2(7) | | Base type of the return type of the method:<br>• int<br>• long<br>• float<br>• double<br>• boolean<br>• byte<br>• char<br>• short<br>• class |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • void |
| RETURN_CLASS | VARCHAR2(4000) | | Actual class name of the return value if the base type is class |
| IS_COMPILED | VARCHAR2(3) | | Indicates whether the Java method has been natively compiled by the JIT compiler (YES) or not (NO) |

**See Also:**

- "DBA_JAVA_METHODS"
- "USER_JAVA_METHODS"

# ALL_JAVA_NCOMPS

ALL_JAVA_NCOMPS displays ncomp-related information about the Java classes accessible to the current user.

### Related Views

- DBA_JAVA_NCOMPS displays ncomp-related information about all Java classes in the database.
- USER_JAVA_NCOMPS displays ncomp-related information about the Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class object |
| NAME | VARCHAR2(4000) | | Name of the Java class object |
| SOURCE | VARCHAR2(4000) | | ncomp source shown in this row |
| INITIALIZER | VARCHAR2(4000) | | ncomp initializer shown in this row |
| LIBRARYFILE | VARCHAR2(4000) | | ncomp library file shown in this row |
| LIBRARY | VARCHAR2(4000) | | ncomp library shown in this row |

**See Also:**

- "DBA_JAVA_NCOMPS"
- "USER_JAVA_NCOMPS"

# ALL_JAVA_RESOLVERS

ALL_JAVA_RESOLVERS displays information about resolvers of the Java classes accessible to the current user.

### Related Views

- DBA_JAVA_RESOLVERS displays information about resolvers of all Java classes in the database.
- USER_JAVA_RESOLVERS displays information about resolvers of the Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class object |
| NAME | VARCHAR2(4000) | | Name of the Java class object |
| TERM_INDEX | NUMBER | | Index of the resolver term in this row |
| PATTERN | VARCHAR2(4000) | | Resolver pattern of the resolver term identified by the TERM_INDEX column |
| SCHEMA | VARCHAR2(64) | | Resolver schema of the resolver term identified by the TERM_INDEX column |

**See Also:**

- "DBA_JAVA_RESOLVERS"
- "USER_JAVA_RESOLVERS"

# ALL_JAVA_THROWS

ALL_JAVA_THROWS displays information about exceptions thrown from methods of the Java classes accessible to the current user.

### Related Views

- DBA_JAVA_THROWS displays information about exceptions thrown from methods of all Java classes in the database.
- USER_JAVA_THROWS displays information about exceptions thrown from methods of the Java classes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Java class |
| NAME | VARCHAR2(4000) | | Name of the Java class |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| METHOD_INDEX | NUMBER | | Index of the throwing method of the exception |
| METHOD_NAME | VARCHAR2(4000) | | Name of the throwing method of the exception |
| EXCEPTION_INDEX | NUMBER | | Index of the exception |
| EXCEPTION_CLASS | VARCHAR2(4000) | | Class of the exception |

**See Also:**

- "DBA_JAVA_THROWS"
- "USER_JAVA_THROWS"

# ALL_JOBS

ALL_JOBS is a synonym for USER_JOBS.

**See Also:**

"USER_JOBS"

# ALL_JOIN_IND_COLUMNS

ALL_JOIN_IND_COLUMNS describes the join conditions of bitmap join indexes accessible to the current user. Bitmap join indexes are indexes built on a child table with an index key containing columns from associated parent tables, where all of the tables are connected through join conditions. There is one row for each join condition.

### Related Views

- DBA_JOIN_IND_COLUMNS describes all join conditions in the database.
- USER_JOIN_IND_COLUMNS describes the join conditions owned by the current user. This view does not display the INDEX_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_OWNER | VARCHAR2(128) | NOT NULL | Owner of the bitmap join index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the bitmap join index |
| INNER_TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the fact table |
| INNER_TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the fact table |
| INNER_TABLE_COLUMN | VARCHAR2(128) | NOT NULL | Name of the fact table join column |
| OUTER_TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the dimension table |
| OUTER_TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the dimension table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OUTER_TABLE_COLUMN | VARCHAR2(128) | NOT NULL | Name of the dimension table join column |

**See Also:**

- "DBA_JOIN_IND_COLUMNS"
- "USER_JOIN_IND_COLUMNS"

# ALL_JSON_COLUMNS

ALL_JSON_COLUMNS provides information on the JavaScript Object Notation (JSON) columns accessible to the current user. Each column that has an IS JSON check constraint in an AND condition and is accessible to the user appears in this view. This view enables a user to find all the JSON columns that are accessible to him or her.

For example, if a check constraint combines the IS JSON condition with another condition using logical condition OR, then the column is not listed in this view. In this case, it is not certain that the data in the column is JSON data. For example, the following constraint does not ensure that the data in column jcol is JSON data:

```
jcol is json OR length(jcol) < 1000
```

### Related Views

- DBA_JSON_COLUMNS provides information on all JSON columns.
- USER_JSON_COLUMNS provides information on the JSON columns for which the user is the owner. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table with the JSON column |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table with the JSON column |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the JSON column |
| FORMAT | VARCHAR2(9) | | Format of the JSON data |
| DATA_TYPE | VARCHAR2(13) | | Data type of the JSON column |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_JSON_COLUMNS"
- "USER_JSON_COLUMNS"
- *Oracle XML DB Developer's Guide* for more information about using JSON with Oracle Database

# ALL_LIBRARIES

ALL_LIBRARIES describes the libraries accessible to the current user.

### Related Views

- DBA_LIBRARIES describes all libraries in the database.
- USER_LIBRARIES describes the libraries owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the library |
| LIBRARY_NAME | VARCHAR2(128) | NOT NULL | Library name |
| FILE_SPEC | VARCHAR2(2000) | | Operating system file specification associated with the library |
| DYNAMIC | VARCHAR2(1) | | Indicates whether the library is dynamically loadable (Y) or not (N) |
| STATUS | VARCHAR2(7) | | Status of the library:<br>• N/A<br>• VALID<br>• INVALID |
| AGENT | VARCHAR2(128) | | Agent of the library |
| LEAF_FILENAME | VARCHAR2(2000) | | Leaf filename of the library |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_LIBRARIES"
- "USER_LIBRARIES"

# ALL_LOB_PARTITIONS

ALL_LOB_PARTITIONS displays the LOB partitions in the tables accessible to the current user.

### Related Views

- DBA_LOB_PARTITIONS displays all LOB partitions in the database.
- USER_LOB_PARTITIONS displays the LOB partitions owned by the current user. This view does not display the TABLE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the LOB column |
| LOB_NAME | VARCHAR2(128) | | Name of the partitioned LOB item |
| PARTITION_NAME | VARCHAR2(128) | | Name of the table partition |
| LOB_PARTITION_NAME | VARCHAR2(128) | | Name of the LOB data partition |
| LOB_INDPART_NAME | VARCHAR2(128) | | Name of the corresponding LOB index partition |
| PARTITION_POSITION | NUMBER | | Position of the LOB data partition within the LOB item |
| COMPOSITE | VARCHAR2(3) | | Indicates whether the partition is composite (YES) or not (NO) |
| CHUNK | NUMBER | | Value of the CHUNK attribute of the LOB data partition |
| PCTVERSION | NUMBER | | Value of the PCTVERSION attribute of the LOB data partition |
| CACHE | VARCHAR2(10) | | Indicates whether and how the LOB data is to be cached in the buffer cache:<br>• YES - LOB data is placed in the buffer cache<br>• NO - LOB data either is not brought into the buffer cache or is brought into the buffer cache and placed at the least recently used end of the LRU list<br>• CACHEREADS - LOB data is brought into the buffer cache only during read operations but not during write operations |
| IN_ROW | VARCHAR2(3) | | Indicates whether the STORAGE IN ROW attribute is enabled for the LOB data partition (YES) or not (NO) |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the LOB data partition |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INITIAL_EXTENT | VARCHAR2(40) | | Size (in bytes) of the initial extent of the LOB data partition, or DEFAULT |
| NEXT_EXTENT | VARCHAR2(40) | | Size (in bytes) of secondary extents of the LOB data partition, or DEFAULT |
| MIN_EXTENTS | VARCHAR2(40) | | Minimum number of extents allowed in the segment of the LOB data partition, or DEFAULT |
| MAX_EXTENTS | VARCHAR2(40) | | Maximum number of extents allowed in the segment of the LOB data partition, or DEFAULT |
| MAX_SIZE | VARCHAR2(40) | | Maximum number of blocks allowed in the segment of the LOB data partition, or DEFAULT |
| RETENTION | VARCHAR2(7) | | Retention option.<br>Possible values for a SecureFiles segment:<br>• NONE<br>• AUTO<br>• MIN<br>• MAX<br>• DEFAULT<br>• INVALID<br>Possible values for a BasicFiles segment:<br>• YES<br>• NO |
| MINRETENTION | VARCHAR2(40) | | Minimum retention duration for a SecureFiles segment, or DEFAULT |
| PCT_INCREASE | VARCHAR2(40) | | Percentage increase in extent size for the LOB data partition, or DEFAULT |
| FREELISTS | VARCHAR2(40) | | Number of process freelists allocated in the segment of the LOB data partition, or DEFAULT |
| FREELIST_GROUPS | VARCHAR2(40) | | Number of freelist groups allocated in the segment of the LOB data partition, or DEFAULT |
| LOGGING | VARCHAR2(7) | | Indicates whether or not changes to the LOB are logged:<br>• NONE - Not specified<br>**See Also:** the *_LOB_SUBPARTITIONS view<br>• YES<br>• NO |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool for the LOB partition blocks:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for partition blocks:<br>• DEFAULT<br>• KEEP |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for partition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| ENCRYPT | VARCHAR2(4) | | Indicates whether or not the LOB is encrypted.<br>Possible values for SecureFiles:<br>• YES<br>• NO<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| COMPRESSION | VARCHAR2(6) | | Level of compression used for this LOB.<br>Possible values for SecureFiles:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• NO - Compression is off<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| DEDUPLICATION | VARCHAR2(15) | | Kind of deduplication used for this LOB.<br>Possible values for SecureFiles:<br>• LOB - Deduplicate<br>• NO - Keep duplicates<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| SECUREFILE | VARCHAR2(3) | | Indicates whether the LOB is SecureFiles (YES) or not (NO) |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the LOB partition segment has been created (YES) or not (NO); N/A indicates that this LOB is subpartitioned and no segment exists at the partition level |

**See Also:**

- "DBA_LOB_PARTITIONS"
- "USER_LOB_PARTITIONS"

# ALL_LOB_SUBPARTITIONS

ALL_LOB_SUBPARTITIONS displays partition-level attributes of the LOB data subpartitions accessible to the current user.

### Related Views

- DBA_LOB_SUBPARTITIONS displays partition-level attributes of all LOB data subpartitions in the database.
- USER_LOB_SUBPARTITIONS displays the LOB subpartitions owned by the current user. This view does not display the TABLE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the LOB column |
| LOB_NAME | VARCHAR2(128) | NOT NULL | Name of the partitioned LOB item |
| LOB_PARTITION_NAME | VARCHAR2(128) | | Name of the LOB data partition to which this LOB data subpartition belongs |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the table subpartition to which this LOB subpartition corresponds |
| LOB_SUBPARTITION_NAME | VARCHAR2(128) | | Name of the LOB subpartition |
| LOB_INDSUBPART_NAME | VARCHAR2(128) | | Name of the corresponding LOB index subpartition |
| SUBPARTITION_POSITION | NUMBER | | Position of the LOB data partition within the LOB item |
| CHUNK | NUMBER | | Value of the CHUNK attribute of the LOB data partition |
| PCTVERSION | NUMBER | NOT NULL | Value of the PCTVERSION attribute of the LOB data partition |
| CACHE | VARCHAR2(10) | | Indicates whether and how the LOB data is to be cached in the buffer cache:<br>• YES - LOB data is placed in the buffer cache<br>• NO - LOB data either is not brought into the buffer cache or is brought into the buffer cache and placed at the least recently used end of the LRU list<br>• CACHEREADS - LOB data is brought into the buffer cache only during read operations but not during write operations |
| IN_ROW | VARCHAR2(3) | | Indicates whether the STORAGE IN ROW attribute of the LOB data partition is enabled (YES) or not (NO) |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace containing the LOB data partition |
| INITIAL_EXTENT | VARCHAR2(40) | | Size in bytes of the initial extent for the LOB data partition |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NEXT_EXTENT | VARCHAR2(40) | | Size in bytes of secondary extents for the LOB data partition |
| MIN_EXTENTS | VARCHAR2(40) | NOT NULL | Minimum number of extents allowed in the segment of the LOB data partition |
| MAX_EXTENTS | VARCHAR2(40) | NOT NULL | Maximum number of extents allowed in the segment of the LOB data partition |
| MAX_SIZE | VARCHAR2(40) | | Maximum number of blocks allowed in the segment of the LOB data partition |
| RETENTION | VARCHAR2(7) | | Retention option.<br>Possible values for a SecureFiles segment:<br>• NONE<br>• AUTO<br>• MIN<br>• MAX<br>• DEFAULT<br>• INVALID<br>Possible values for a BasicFiles segment:<br>• YES<br>• NO |
| MINRETENTION | VARCHAR2(40) | | Minimum retention duration for a SecureFiles segment |
| PCT_INCREASE | VARCHAR2(40) | | Percentage increase in extent size for the LOB data partition |
| FREELISTS | VARCHAR2(40) | | Number of process freelists allocated in the segment of the LOB data partition |
| FREELIST_GROUPS | VARCHAR2(40) | | Number of freelist groups allocated in the segment of the LOB data partition |
| LOGGING | VARCHAR2(3) | | Indicates whether or not changes to the LOB are logged:<br>• YES<br>• NO |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool to be used for the LOB data partition blocks:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for subpartition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for subpartition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| ENCRYPT | VARCHAR2(4) | | Indicates whether or not the LOB is encrypted.<br>Possible values for SecureFiles:<br>• YES<br>• NO<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| COMPRESSION | VARCHAR2(6) | | Level of compression used for this LOB.<br>Possible values for SecureFiles:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• NO - Compression is off<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| DEDUPLICATION | VARCHAR2(15) | | Kind of deduplication used for this LOB.<br>Possible values for SecureFiles:<br>• LOB - Deduplicate<br>• NO - Keep duplicates<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| SECUREFILE | VARCHAR2(3) | | Indicates whether the LOB is SecureFiles (YES) or not (NO) |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the LOB subpartition segment has been created (YES) or not (NO); N/A indicates that this LOB is not subpartitioned |

**See Also:**

- "DBA_LOB_SUBPARTITIONS"
- "USER_LOB_SUBPARTITIONS"

# ALL_LOB_TEMPLATES

ALL_LOB_TEMPLATES describes the LOB subpartition templates accessible to the current user.

### Related Views

- DBA_LOB_TEMPLATES describes all LOB subpartition templates in the database.
- USER_LOB_TEMPLATES describes the LOB subpartition templates owned by the current user. This view does not display the USER_NAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_NAME | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| LOB_COL_NAME | VARCHAR2(4000) | | Name of the LOB column |
| SUBPARTITION_NAME | VARCHAR2(132) | NOT NULL | Name of the subpartition |
| LOB_SEGMENT_NAME | VARCHAR2(132) | NOT NULL | Name of the LOB segment |
| TABLESPACE_NAME | VARCHAR2(30) | | Tablespace name of the subpartition |

**See Also:**

- "DBA_LOB_TEMPLATES"
- "USER_LOB_TEMPLATES"

# ALL_LOBS

ALL_LOBS displays the large objects (LOBs) contained in tables accessible to the current user. LOBs include binary large objects (BLOBs) and character large objects (CLOBs). Binary files (BFILEs) are stored outside the database, so they are not displayed by this view or the related views.

### Related Views

- DBA_LOBS describes all LOBs in the database.
- USER_LOBS describes the LOBs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the object containing the LOB |
| TABLE_NAME | VARCHAR2(128) | | Name of the object containing the LOB |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the LOB column or attribute |
| SEGMENT_NAME | VARCHAR2(128) | | Name of the LOB segment |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the LOB segment |
| INDEX_NAME | VARCHAR2(128) | | Name of the LOB index |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CHUNK | NUMBER | | Size (in bytes) of the LOB chunk as a unit of allocation or manipulation |
| PCTVERSION | NUMBER | | Maximum percentage of the LOB space used for versioning |
| RETENTION | NUMBER | | Maximum time duration for versioning of the LOB space |
| FREEPOOLS | NUMBER | | Number of freepools for this LOB segment |
| CACHE | VARCHAR2(10) | | Indicates whether and how the LOB data is to be cached in the buffer cache:<br>• YES - LOB data is placed in the buffer cache<br>• NO - LOB data either is not brought into the buffer cache or is brought into the buffer cache and placed at the least recently used end of the LRU list<br>• CACHEREADS - LOB data is brought into the buffer cache only during read operations but not during write operations |
| LOGGING | VARCHAR2(7) | | Indicates whether or not changes to the LOB are logged:<br>• NONE - Not specified<br>**See Also:** the *_LOB_SUBPARTITIONS view<br>• YES<br>• NO |
| ENCRYPT | VARCHAR2(4) | | Indicates whether or not the LOB is encrypted.<br>Possible values for SecureFiles:<br>• YES<br>• NO<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| COMPRESSION | VARCHAR2(6) | | Level of compression used for this LOB.<br>Possible values for SecureFiles:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• NO - Compression is off<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| DEDUPLICATION | VARCHAR2(15) | | Kind of deduplication used for this LOB.<br>Possible values for SecureFiles:<br>• LOB - Deduplicate<br>• NO - Keep duplicates<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| IN_ROW | VARCHAR2(3) | | Indicates whether some LOBs are stored inline with the base row (YES) or not (NO). For |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | partitioned objects, refer to the *_LOB_PARTITIONS and *_PART_LOBS views. |
| FORMAT | VARCHAR2(15) | | Indicates whether the LOB storage format depends on the endianness of the platform:<br>• NOT APPLICABLE<br>• ENDIAN SPECIFIC<br>• ENDIAN NEUTRAL |
| PARTITIONED | VARCHAR2(3) | | Indicates whether the LOB column is in a partitioned table (YES) or not (NO) |
| SECUREFILE | VARCHAR2(3) | | Indicates whether the LOB is SecureFiles (YES) or not (NO) |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the LOB segment has been created (YES) or not (NO) |
| RETENTION_TYPE | VARCHAR2(7) | | Type of retention used for this LOB.<br>Possible values for SecureFiles:<br>• NONE<br>• AUTO<br>• MIN<br>• MAX<br>• DEFAULT<br>• INVALID<br>Possible values for BasicFiles:<br>• YES<br>• NO |
| RETENTION_VALUE | NUMBER | | Minimum retention time (in seconds). This column is only meaningful for SecureFiles with RETENTION_TYPE set to MIN. |

**See Also:**

- "DBA_LOBS"
- "USER_LOBS"

# ALL_LOG_GROUP_COLUMNS

ALL_LOG_GROUP_COLUMNS describes columns that are accessible to the current user and that are specified in log groups.

### Related Views

- DBA_LOG_GROUP_COLUMNS describes all columns in the database that are specified in log groups.
- USER_LOG_GROUP_COLUMNS describes columns that are owned by the current user and that are specified in log groups.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the log group definition |
| LOG_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the log group definition |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table in which the log group is defined |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column or attribute of the object type column specified in the log group definition |
| POSITION | NUMBER | | Original position of the column or attribute in the definition of the object |
| LOGGING_PROPERTY | VARCHAR2(6) | | Indicates whether the column or attribute would be supplementally logged (LOG) or not (NO LOG) |

**See Also:**

- "DBA_LOG_GROUP_COLUMNS"
- "USER_LOG_GROUP_COLUMNS"

# ALL_LOG_GROUPS

ALL_LOG_GROUPS describes the log group definitions on the tables accessible to the current user.

### Related Views

- DBA_LOG_GROUPS describes the log group definitions on all tables in the database.
- USER_LOG_GROUPS describes the log group definitions on the tables owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the log group definition |
| LOG_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the log group definition |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table on which the log group is defined |
| LOG_GROUP_TYPE | VARCHAR2(28) | | Type of the log group:<br>• PRIMARY KEY LOGGING<br>• UNIQUE KEY LOGGING<br>• FOREIGN KEY LOGGING<br>• ALL COLUMN LOGGING<br>• USER LOG GROUP |
| ALWAYS | VARCHAR2(11) | | Y indicates the log group is logged any time a row is updated; N indicates the log group is logged any time a member column is updated |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GENERATED | VARCHAR2(14) | | Indicates whether the name of the supplemental log group was system generated (GENERATED NAME) or not (USER NAME) |

**See Also:**

- "DBA_LOG_GROUPS"
- "USER_LOG_GROUPS"

# ALL_MEASURE_FOLDER_CONTENTS

ALL_MEASURE_FOLDER_CONTENTS describes the contents of the OLAP measure folders accessible to the current user.

### Related Views

- DBA_MEASURE_FOLDER_CONTENTS describes the contents of all OLAP measure folders in the database.
- USER_MEASURE_FOLDER_CONTENTS describes the contents of the OLAP measure folders owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the measure folder |
| MEASURE_FOLDER_NAME | VARCHAR2(128) | NOT NULL | Name of a measure folder |
| CUBE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the cube |
| CUBE_NAME | VARCHAR2(128) | NOT NULL | Name of a cube included in the measure folder |
| MEASURE_NAME | VARCHAR2(128) | NOT NULL | Name of a measure in the cube |
| ORDER_NUM | NUMBER | NOT NULL | Order number of the measure in the folder |

**See Also:**

- "DBA_MEASURE_FOLDER_CONTENTS"
- "USER_MEASURE_FOLDER_CONTENTS"

# ALL_MEASURE_FOLDER_SUBFOLDERS

ALL_MEASURE_FOLDER_SUBFOLDERS describes the OLAP measure folders contained within the OLAP measure folders accessible to the user.

### Related Views

- DBA_MEASURE_FOLDER_SUBFOLDERS describes the OLAP measure folders contained within the database OLAP measure folders.
- USER_MEASURE_FOLDER_SUBFOLDERS describes the OLAP measure folders contained within the OLAP measure folders owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the OLAP measure folder that contains a subfolder |
| MEASURE_FOLDER_NAME | VARCHAR2(128) | NOT NULL | Name of the OLAP measure folder that contains a subfolder |
| MEASURE_SUBFOLDER_OWNER | VARCHAR2(128) | NOT NULL | Owner of the OLAP measure folder subfolder |
| MEASURE_SUBFOLDER_NAME | VARCHAR2(128) | NOT NULL | Name of the owning OLAP measure folder subfolder |

**See Also:**

- "DBA_MEASURE_FOLDER_SUBFOLDERS"
- "USER_MEASURE_FOLDER_SUBFOLDERS"

# ALL_MEASURE_FOLDERS

ALL_MEASURE_FOLDERS describes the OLAP measure folders accessible to the current user.

### Related Views

- DBA_MEASURE_FOLDERS describes all OLAP measure folders in the database.
- USER_MEASURE_FOLDERS describes the OLAP measure folders owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the measure folder |
| MEASURE_FOLDER_NAME | VARCHAR2(128) | NOT NULL | Name of a measure folder |
| MEASURE_FOLDER_ID | NUMBER | NOT NULL | ID of a measure folder |
| DESCRIPTION | NVARCHAR2(300) | | Description of the measure folder in the session language |

**See Also:**

- "DBA_MEASURE_FOLDERS"
- "USER_MEASURE_FOLDERS"

# ALL_METADATA_PROPERTIES

ALL_METADATA_PROPERTIES describes OLAP metadata properties in the database that are accessible to the current user.

### Related Views

- DBA_METADATA_PROPERTIES describes OLAP metadata properties in the database.
- USER_METADATA_PROPERTIES describes OLAP metadata properties in the current user's schema. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the OLAP metadata property |
| OWNING_OBJECT_ID | NUMBER | | Dictionary ID of the OLAP metadata property owner |
| OWNING_TYPE | VARCHAR2(23) | | Owning type of the OLAP metadata property |
| PROPERTY_ID | NUMBER | | Dictionary Id of the OLAP metadata property |
| PROPERTY_KEY | VARCHAR2(128) | | Key of the OLAP metadata property |
| PROPERTY_VALUE | CLOB | | Value of the OLAP metadata property |
| PROPERTY_ORDER | NUMBER | | Order number of the OLAP metadata property |

**See Also:**

- "DBA_METADATA_PROPERTIES"
- "USER_METADATA_PROPERTIES"

# ALL_METHOD_PARAMS

ALL_METHOD_PARAMS describes the method parameters of the object types accessible to the current user.

### Related Views

- DBA_METHOD_PARAMS describes the method parameters of all object types in the database.
- USER_METHOD_PARAMS describes the method parameters of the object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the type |
| METHOD_NAME | VARCHAR2(128) | NOT NULL | Name of the method |
| METHOD_NO | NUMBER | NOT NULL | For an overloaded method, a number distinguishing this method from others of the same. Do not confuse this number with the object ID. |
| PARAM_NAME | VARCHAR2(128) | NOT NULL | Name of the parameter |
| PARAM_NO | NUMBER | NOT NULL | Parameter number (position) |
| PARAM_MODE | VARCHAR2(6) | | Mode of the parameter (IN, OUT, IN OUT) |
| PARAM_TYPE_MOD | VARCHAR2(7) | | Whether this parameter is a REF to another object |
| PARAM_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the parameter |
| PARAM_TYPE_NAME | VARCHAR2(128) | | Name of the type of the parameter |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Whether the character set or the method is fixed-length character set (CHAR_CS) or fixed-length national character set (NCHAR_CS), or a particular character set specified by the user |

**See Also:**

- "DBA_METHOD_PARAMS"
- "USER_METHOD_PARAMS"

# ALL_METHOD_RESULTS

ALL_METHOD_RESULTS describes the method results of the object types accessible to the current user.

### Related Views

- DBA_METHOD_RESULTS describes the method results of all object types in the database.
- USER_METHOD_RESULTS describes the method results of the object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the type |
| METHOD_NAME | VARCHAR2(128) | NOT NULL | Name of the method |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| METHOD_NO | NUMBER | NOT NULL | For an overloaded method, a number distinguishing this method from others of the same. Do not confuse this number with the object ID. |
| RESULT_TYPE_MOD | VARCHAR2(7) | | Whether this result is a REF to another object |
| RESULT_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the result |
| RESULT_TYPE_NAME | VARCHAR2(128) | | Name of the type of the result |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Whether the character set or the method is fixed-length character set (CHAR_CS) or fixed-length national character set (NCHAR_CS), or a particular character set specified by the user |

**See Also:**

- "DBA_METHOD_RESULTS"
- "USER_METHOD_RESULTS"

# ALL_MINING_MODEL_ATTRIBUTES

ALL_MINING_MODEL_ATTRIBUTES describes the attributes of the mining models accessible to the current user. Only the attributes in the model signature are included in this view. The attributes in the model signature correspond to the columns in the training data that were used to build the model.

Mining models are schema objects created by Oracle Data Mining.

### Related Views

- DBA_MINING_MODEL_ATTRIBUTES describes the attributes of all mining models in the database.
- USER_MINING_MODEL_ATTRIBUTES describes the attributes of the mining models owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the mining model |
| MODEL_NAME | VARCHAR2(128) | NOT NULL | Name of the mining model |
| ATTRIBUTE_NAME | VARCHAR2(128) | NOT NULL | Name of the attribute |
| ATTRIBUTE_TYPE | VARCHAR2(11) | | Logical type of the attribute. The type is identified during the model build or apply process:<br>• NUMERICAL - Numeric data<br>• CATEGORICAL - Character data<br>• TEXT - Unstructured text data |
| DATA_TYPE | VARCHAR2(106) | | Data type of the attribute |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DATA_LENGTH | NUMBER | | Length of the data type |
| DATA_PRECISION | NUMBER | | Precision of a fixed point number. Precision, which is the total number of significant decimal digits, is represented as *p* in the data type NUMBER(p,s). |
| DATA_SCALE | NUMBER | | Scale of a fixed point number. Scale, which is the number of digits from the decimal to the least significant digit, is represented as *s* in the data type NUMBER(p,s). |
| USAGE_TYPE | VARCHAR2(8) | | Indicates whether the attribute was used to construct the model (ACTIVE) or not (INACTIVE). Some attributes may be eliminated by transformations or algorithmic processing. The *_MINING_MODEL_ATTRIBUTES view only lists the attributes used by the model, therefore the value of this column is always ACTIVE. |
| TARGET | VARCHAR2(3) | | Indicates whether the attribute is the target of a predictive model (YES) or not (NO). The target describes the result that is produced when the model is applied. |
| ATTRIBUTE_SPEC | VARCHAR2(4000) | | One or more keywords that identify special treatment for the attribute during model build. Values are:<br>• FORCE_IN - (GLM only) When feature selection is enabled, forces the inclusion of the attribute in the model build. Feature selection is disabled by default. If the model is not using GLM with feature selection enabled, this value is ignored.<br>• NOPREP - When ADP is on, prevents automatic transformation of the attribute. If ADP is not on, this value is ignored.<br>• TEXT - Causes the attribute to be treated as unstructured text data. The TEXT value supports three subsettings: POLICY_NAME, MAX_FEATURES, and TOKEN_TYPE. Subsettings are specified as name:value pairs within parentheses. For example: (POLICY_NAME:mypolicy) (MAX_FEATURES:2000) (TOKEN_TYPE:THEME). See *Oracle Data Mining User's Guide* for details.<br>• NULL - The ATTRIBUTE_SPEC for this attribute is NULL.<br>ATTRIBUTE_SPEC is a parameter to the PL/SQL procedure DBMS_DATA_MINING_TRANSFORM.SET_TRANSFORM. See *Oracle Database PL/SQL Packages and Types Reference* for details. |

**See Also:**

- "DBA_MINING_MODEL_ATTRIBUTES"
- "USER_MINING_MODEL_ATTRIBUTES"
- *Oracle Data Mining User's Guide* for information about model attributes

# ALL_MINING_MODEL_SETTINGS

ALL_MINING_MODEL_SETTINGS describes the settings of the mining models accessible to the current user.

Mining models are schema objects created by Oracle Data Mining.

### Related Views

- DBA_MINING_MODEL_SETTINGS describes the settings of all mining models in the database.
- USER_MINING_MODEL_SETTINGS describes the settings of the mining models owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the mining model |
| MODEL_NAME | VARCHAR2(128) | NOT NULL | Name of the mining model |
| SETTING_NAME | VARCHAR2(30) | NOT NULL | Name of the setting |
| SETTING_VALUE | VARCHAR2(4000) | | Value of the setting |
| SETTING_TYPE | VARCHAR2(7) | | Indicates whether the default value (DEFAULT) or a user-specified value (INPUT) is used by the model |

**See Also:**

- "DBA_MINING_MODEL_SETTINGS"
- "USER_MINING_MODEL_SETTINGS"
- *Oracle Database PL/SQL Packages and Types Reference* for descriptions of model settings

# ALL_MINING_MODELS

ALL_MINING_MODELS describes the mining models accessible to the current user.

Mining models are schema objects created by Oracle Data Mining.

### Related Views

- DBA_MINING_MODELS describes all mining models in the database.
- USER_MINING_MODELS describes the mining models owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the mining model |
| MODEL_NAME | VARCHAR2(128) | NOT NULL | Name of the mining model |
| MINING_FUNCTION | VARCHAR2(30) | | Function of the mining model. The function identifies the class of problems that can be solved by this model. The mining function is specified when the model is built:<br>• CLASSIFICATION<br>• REGRESSION<br>• CLUSTERING<br>• FEATURE_EXTRACTION<br>• ASSOCIATION_RULES<br>• ATTRIBUTE_IMPORTANCE |
| ALGORITHM | VARCHAR2(30) | | Algorithm used by the model. Each mining function has a default algorithm. The default can be overridden with a model setting (see *_MINING_MODEL_SETTINGS):<br>• NAIVE_BAYES<br>• DECISION_TREE<br>• SUPPORT_VECTOR_MACHINES<br>• KMEANS<br>• O_CLUSTER<br>• NONNEGATIVE_MATRIX_FACTOR<br>• GENERALIZED_LINEAR_MODEL<br>• APRIORI_ASSOCIATION_RULES<br>• MINIMUM_DESCRIPTION_LENGTH<br>• EXPECTATION_MAXIMIZATION<br>• SINGULAR_VALUE_DECOMP |
| CREATION_DATE | DATE | NOT NULL | Date that the model was created |
| BUILD_DURATION | NUMBER | | Time (in seconds) of the model build process |
| MODEL_SIZE | NUMBER | | Size of the model (in megabytes) |
| COMMENTS | VARCHAR2(4000) | | Comment applied to the model with a SQL COMMENT statement |

**See Also:**

- "DBA_MINING_MODELS"
- "USER_MINING_MODELS"
- *Oracle Data Mining User's Guide* for information about mining model schema objects
- *Oracle Data Mining Concepts* for an introduction to Data Mining

# ALL_MVIEW_AGGREGATES

ALL_MVIEW_AGGREGATES describes the grouping functions (aggregate operations) that appear in the SELECT list of materialized aggregate views accessible to the current user.

### Related Views

- DBA_MVIEW_AGGREGATES describes all such grouping functions defined for all materialized views in the database.
- USER_MVIEW_AGGREGATES describes all such grouping functions defined for all materialized views owned by the current user.

**Note:**

All three views exclude materialized views that reference remote tables or that include references to a nonstatic value such as SYSDATE or USER. These views also exclude materialized views that were created as "snapshots" before Oracle8*i* and that were never altered to enable query rewrite.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| POSITION_IN_SELECT | NUMBER | NOT NULL | Ordinal position of this aggregation within the SELECT list. For the position of nonaggregate elements of the select list, see "ALL_MVIEW_KEYS". |
| CONTAINER_COLUMN | VARCHAR2(128) | NOT NULL | Name of this column in the container table |
| AGG_FUNCTION | VARCHAR2(8) | | Aggregation function |
| DISTINCTFLAG | VARCHAR2(1) | | Indicates whether this aggregation is distinct (Y) or not (N) |
| MEASURE | LONG | | SQL text of the measure, excluding the aggregation function. Equal to * for COUNT(*). |

**See Also:**

- "DBA_MVIEW_AGGREGATES"
- "USER_MVIEW_AGGREGATES"

# ALL_MVIEW_ANALYSIS

ALL_MVIEW_ANALYSIS describes the materialized views accessible to the current user. It provides additional information for analysis by applications. Minimal information is displayed for materialized views that do not support query rewrite (such as materialized views with remote master tables or nondeterministic functions).

### Related Views

- DBA_MVIEW_ANALYSIS describes all such materialized views in the database.
- USER_MVIEW_ANALYSIS describes all such materialized views owned by the current user.

**Note:**

All of the information in these views is also displayed in ALL_MVIEWS and its related views. Oracle recommends that you refer to ALL_MVIEWS for this information instead of these views.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| MVIEW_TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the container table (see next column) |
| CONTAINER_NAME | VARCHAR2(128) | | Name of the internal container in which the materialized view data is held. Normally this is the same as MVIEW_NAME. For materialized views created before Oracle8*i*, Oracle Database attaches the 6-byte prefix SNAP$_. If MVIEW_NAME has more than 19 bytes, then Oracle Database truncates the name to 19 bytes and adds a 4-byte sequence number as a suffix to produce a nonambiguous CONTAINER_NAME. |
| LAST_REFRESH_SCN | NUMBER | | System change number (SCN) of the last refresh operation |
| LAST_REFRESH_DATE | DATE | | SYSDATE of the last refresh |
| REFRESH_METHOD | VARCHAR2(8) | | Default refresh method:<br>• FORCE<br>• FAST<br>• COMPLETE<br>• NEVER |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SUMMARY | VARCHAR2(1) | | Indicates whether this materialized view includes a GROUP BY clause or aggregation (Y) or not (N) |
| FULLREFRESHTIM | NUMBER | | Approximate refresh time, in seconds, for full refresh |
| INCREFRESHTIM | NUMBER | | Approximate refresh time, in seconds, for fast refresh |
| CONTAINS_VIEWS | VARCHAR2(1) | | Indicates whether this materialized view contains a view in its definition (Y) or not (N) |
| UNUSABLE | VARCHAR2(1) | | Indicates whether this materialized view is UNUSABLE (inconsistent data) (Y) or not (N). A materialized view can be UNUSABLE if a system failure occurs during a full refresh. |
| RESTRICTED_SYNTAX | VARCHAR2(1) | | Indicates whether this materialized view had a restriction in its defining query that limits the use of query rewrite (Y) or not (N). More complete information is provided by the REWRITE_CAPABILITY column of the *_MVIEWS view. |
| INC_REFRESHABLE | VARCHAR2(1) | | Indicates whether this materialized view can be fast refreshed (Y) or not (N) |
| KNOWN_STALE | VARCHAR2(1) | | Indicates whether the data contained in the materialized view is known to be inconsistent with the master table data because that has been updated since the last successful refresh (Y) or not (N) |
| INVALID | VARCHAR2(1) | | Indicates whether this materialized view is in an invalid state (inconsistent metadata) (Y) or not (N) |
| REWRITE_ENABLED | VARCHAR2(1) | | Indicates whether this materialized view is currently enabled for query rewrite (Y) or not (N) |
| QUERY_LEN | NUMBER | | Length (in bytes) of the query field |
| QUERY | LONG | | SELECT expression of the materialized view definition |
| REVISION | NUMBER | NOT NULL | Reserved for internal use |

**See Also:**

- "DBA_MVIEW_ANALYSIS"
- "USER_MVIEW_ANALYSIS"

# ALL_MVIEW_COMMENTS

ALL_MVIEW_COMMENTS displays comments on the materialized views accessible to the current user.

### Related Views

- DBA_MVIEW_COMMENTS displays comments on the materialized views in the database.
- USER_MVIEW_COMMENTS displays comments on the materialized views owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| COMMENTS | VARCHAR2(4000) | | Comment on the materialized view |

**See Also:**

- "DBA_MVIEW_COMMENTS"
- "USER_MVIEW_COMMENTS"

# ALL_MVIEW_DETAIL_PARTITION

ALL_MVIEW_DETAIL_PARTITION displays the freshness information of the materialized views, with respect to a PCT detail partition, accessible to the current user.

### Related Views

- DBA_MVIEW_DETAIL_PARTITION displays freshness information for all materialized views in the database, with respect to a PCT detail partition.
- USER_MVIEW_DETAIL_PARTITION displays freshness information for all materialized views, with respect to a PCT detail partition, owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| DETAILOBJ_OWNER | VARCHAR2(128) | NOT NULL | Owner of the detail object |
| DETAILOBJ_NAME | VARCHAR2(128) | NOT NULL | Name of the detail object |
| DETAIL_PARTITION_NAME | VARCHAR2(128) | | Name of the detail object partition |
| DETAIL_PARTITION_POSITION | NUMBER | | Position of the detail object partition |
| FRESHNESS | CHAR(5) | | Freshness state (FRESH, STALE, UNKNOWN, NA, and so on) |

**See Also:**

- "DBA_MVIEW_DETAIL_PARTITION"
- "USER_MVIEW_DETAIL_PARTITION"

# ALL_MVIEW_DETAIL_RELATIONS

ALL_MVIEW_DETAIL_RELATIONS describes the named detail relations that are either specified in the FROM list of the subquery that defines a materialized view accessible to the current user, or that are indirectly referenced through views in that FROM list. Inline views in the materialized view definition are not represented in this view or the related views.

### Related Views

- DBA_MVIEW_DETAIL_RELATIONS describes all such detail relations defined for all materialized views in the database.
- USER_MVIEW_DETAIL_RELATIONS describes such detail relations defined for all materialized views owned by the current user.

**Note:**

All three views exclude materialized views that reference remote tables or that includes references to a nonstatic value such as SYSDATE or USER. These views also exclude materialized views that were created as *snapshots* before Oracle8*i* and that were never altered to enable query rewrite.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| DETAILOBJ_OWNER | VARCHAR2(128) | NOT NULL | Detail object owner |
| DETAILOBJ_NAME | VARCHAR2(128) | NOT NULL | Detail object name (that is, the name of a table or view) |
| DETAILOBJ_TYPE | VARCHAR2(9) | | Detail object type:<br>• TABLE<br>• VIEW<br>• SNAPSHOT<br>• CONTAINER<br>• UNDEFINED |
| DETAILOBJ_ALIAS | VARCHAR2(128) | | Implicit or explicit alias for detail relation |
| DETAILOBJ_PCT | VARCHAR2(1) | | Indicates whether the detail object PCT is supported (Y) or not (N) |
| NUM_FRESH_PCT_PARTITIONS | NUMBER | | Number of fresh PCT partitions |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NUM_STALE_PCT_PARTITIONS | NUMBER | | Number of stale PCT partitions |

**See Also:**

- "DBA_MVIEW_DETAIL_RELATIONS"
- "USER_MVIEW_DETAIL_RELATIONS"

# ALL_MVIEW_DETAIL_SUBPARTITION

ALL_MVIEW_DETAIL_SUBPARTITION displays the freshness information of the materialized views, with respect to a PCT detail subpartition, accessible to the current user.

### Related Views

- DBA_MVIEW_DETAIL_SUBPARTITION displays freshness information for all materialized views in the database, with respect to a PCT detail subpartition.
- USER_MVIEW_DETAIL_SUBPARTITION displays freshness information for all materialized views, with respect to a PCT detail subpartition, owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| DETAILOBJ_OWNER | VARCHAR2(128) | NOT NULL | Owner of the detail object |
| DETAILOBJ_NAME | VARCHAR2(128) | NOT NULL | Name of the detail object |
| DETAIL_PARTITION_NAME | VARCHAR2(128) | | Name of the detail object partition |
| DETAIL_SUBPARTITION_NAME | VARCHAR2(128) | | Name of the detail object subpartition |
| DETAIL_SUBPARTITION_POSITION | NUMBER | | Position of the detail object subpartition |
| FRESHNESS | CHAR(5) | | Freshness state (FRESH, STALE, UNKNOWN, NA, and so on) |

**See Also:**

- "DBA_MVIEW_DETAIL_SUBPARTITION"
- "USER_MVIEW_DETAIL_SUBPARTITION"

# ALL_MVIEW_JOINS

ALL_MVIEW_JOINS describes joins between two columns in the WHERE clause of the subquery that defines a materialized view accessible to the current user.

### Related Views

- DBA_MVIEW_JOINS describes all such joins for all materialized views in the database.
- USER_MVIEW_JOINS describes such joins for all materialized views owned by the current user.

**Note:**

All three views exclude materialized views that reference remote tables or that includes references to a nonstatic value such as SYSDATE or USER. These views also exclude materialized views that were created as "snapshots" before Oracle8*i* and that were never altered to enable query rewrite.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Materialized view name |
| DETAILOBJ1_OWNER | VARCHAR2(128) | NOT NULL | Owner of the first object in the join |
| DETAILOBJ1_RELATION | VARCHAR2(128) | NOT NULL | Name of the first object in the join[1] |
| DETAILOBJ1_COLUMN | VARCHAR2(128) | NOT NULL | Join column of the first object in the join[1] |
| OPERATOR | CHAR(1) | | Join operator[1] |
| OPERATOR_TYPE | VARCHAR2(1) | | Indicates whether the join is an inner join (I) or the DETAILOBJ1 table is the left side of an outer join (L)[1] |
| DETAILOBJ2_OWNER | VARCHAR2(128) | NOT NULL | Owner of the second object in the join[1] |
| DETAILOBJ2_RELATION | VARCHAR2(128) | NOT NULL | Name of the second object in the join[1] |
| DETAILOBJ2_COLUMN | VARCHAR2(128) | NOT NULL | Join column of the second object in the join[1] |

[1] These rows relate only to materialized join views and materialized aggregate views. They describe the two detail objects of a materialized view join.

**See Also:**

- "DBA_MVIEW_JOINS"
- "USER_MVIEW_JOINS"

# ALL_MVIEW_KEYS

ALL_MVIEW_KEYS describes the columns or expressions in the SELECT list upon which materialized views accessible to the current user are based.

### Related Views

- DBA_MVIEW_KEYS describes such columns and expressions for all materialized views in the database.
- USER_MVIEW_KEYS describes such columns and expressions for all materialized views owned by the current user.

**Note:**

All three views exclude materialized views that reference remote tables or that includes references to a nonstatic value such as SYSDATE or USER. These views also exclude materialized views that were created as *snapshots* before Oracle8*i* and that were never altered to enable query rewrite.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Materialized view name |
| POSITION_IN_SELECT | NUMBER | NOT NULL | Ordinal position of this key within the SELECT list |
| CONTAINER_COLUMN | VARCHAR2(128) | NOT NULL | Name of the column in the container table |
| DETAILOBJ_OWNER | VARCHAR2(128) | NOT NULL | Detail object owner |
| DETAILOBJ_NAME | VARCHAR2(128) | NOT NULL | Detail object name (for example, the name of a table or view) |
| DETAILOBJ_ALIAS | VARCHAR2(128) | | Implicit or explicit alias for detail relation |
| DETAILOBJ_TYPE | VARCHAR2(5) | | Detail object type:<br>• TABLE<br>• VIEW |
| DETAILOBJ_COLUMN | VARCHAR2(128) | NOT NULL | Name of the detail relation column |

**See Also:**

- "DBA_MVIEW_KEYS"
- "USER_MVIEW_KEYS"

# ALL_MVIEW_LOGS

ALL_MVIEW_LOGS describes all materialized view logs accessible to the current user.

### Related Views

- DBA_MVIEW_LOGS describes all materialized view logs in the database.
- USER_MVIEW_LOGS describes all materialized view logs owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOG_OWNER | VARCHAR2(128) | | Owner of the materialized view log |
| MASTER | VARCHAR2(128) | | Name of the master table or master materialized view whose changes are logged |
| LOG_TABLE | VARCHAR2(128) | | Name of the table where the changes to the master table or master materialized view are logged |
| LOG_TRIGGER | VARCHAR2(128) | | Obsolete with Oracle8*i* and later. Set to NULL. Formerly, this parameter was an after-row trigger on the master which inserted rows into the log. |
| ROWIDS | VARCHAR2(3) | | Indicates whether rowid information is recorded (YES) or not (NO) |
| PRIMARY_KEY | VARCHAR2(3) | | Indicates whether primary key information is recorded (YES) or not (NO) |
| OBJECT_ID | VARCHAR2(3) | | Indicates whether object identifier information in an object table is recorded (YES) or not (NO) |
| FILTER_COLUMNS | VARCHAR2(3) | | Indicates whether filter column information is recorded (YES) or not (NO) |
| SEQUENCE | VARCHAR2(3) | | Indicates whether the sequence value, which provides additional ordering information, is recorded (YES) or not (NO) |
| INCLUDE_NEW_VALUES | VARCHAR2(3) | | Indicates whether both old and new values are recorded (YES) or old values are recorded but new values are not recorded (NO) |
| PURGE_ASYNCHRONOUS | VARCHAR2(3) | | Indicates whether the materialized view log is purged asynchronously (YES) or not (NO) |
| PURGE_DEFERRED | VARCHAR2(3) | | Indicates whether the materialized view log is purged in a deferred manner (YES) or not (NO) |
| PURGE_START | DATE | | For deferred purge, the purge start date |
| PURGE_INTERVAL | VARCHAR2(200) | | For deferred purge, the purge interval |
| LAST_PURGE_DATE | DATE | | Date of the last purge |
| LAST_PURGE_STATUS | NUMBER | | Status of the last purge (error code or 0 for success) |
| NUM_ROWS_PURGED | NUMBER | | Number of rows purged in the last purge |
| COMMIT_SCN_BASED | VARCHAR2(3) | | Indicates whether the materialized view log is commit SCN-based (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STAGING_LOG | VARCHAR2(3) | | Indicates whether the materialized view log is a staging log for synchronous refresh (YES) or not (NO) |

**See Also:**

- "DBA_MVIEW_LOGS"
- "USER_MVIEW_LOGS"

# ALL_MVIEW_REFRESH_TIMES

ALL_MVIEW_REFRESH_TIMES describes refresh times of the materialized views accessible to the current user.

### Related Views

- DBA_MVIEW_REFRESH_TIMES describes refresh times of all materialized views in the database.
- USER_MVIEW_REFRESH_TIMES describes refresh times of the materialized views owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| MASTER_OWNER | VARCHAR2(128) | | Owner of the master table |
| MASTER | VARCHAR2(128) | | Name of the For doc bug 14120613, added description LAST_REFRESH_END_TIME column in ALL_MVIEWS. The column description was provided in "Re: What description should I use for the new LAST_REFRESH_END_TIME column in [ALL\|DBA]_MVIEWS?" August 30, 2013 11:31 AM email from tsae-feng.yu.master table |
| LAST_REFRESH | DATE | | SYSDATE from the master site at the time of the last refresh |

**See Also:**

- "DBA_MVIEW_REFRESH_TIMES"
- "USER_MVIEW_REFRESH_TIMES"

# ALL_MVIEWS

ALL_MVIEWS describes all materialized views accessible to the current user.

### Related views

- DBA_MVIEWS describes all materialized views in the database.
- USER_MVIEWS describes all materialized views owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Schema in which the materialized view was created |
| MVIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| CONTAINER_NAME | VARCHAR2(128) | NOT NULL | Name of the container in which the materialized view's data is held. Normally this is the same as MVIEW_NAME. For materialized views created before Oracle8*i*, Oracle Database attaches the 6-byte prefix SNAP$_. If MVIEW_NAME has more than 19 bytes, then Oracle Database truncates the name to 19 bytes and may add a 4-byte sequence number as a suffix to produce a nonambiguous CONTAINER_NAME. |
| QUERY | LONG | | Query that defines the materialized view |
| QUERY_LEN | NUMBER(38) | | Length (in bytes) of the defining query |
| UPDATABLE | VARCHAR2(1) | | Indicates whether the materialized view is updatable (Y) or not (N) |
| UPDATE_LOG | VARCHAR2(128) | | For updatable materialized views, the filename of the update log |
| MASTER_ROLLBACK_SEG | VARCHAR2(128) | | Rollback segment for the master site or the master materialized view site |
| MASTER_LINK | VARCHAR2(128) | | Database link for the master site or the master materialized view site |
| REWRITE_ENABLED | VARCHAR2(1) | | Indicates whether rewrite is enabled (Y) or not (N) |
| REWRITE_CAPABILITY | VARCHAR2(9) | | Indicates whether the materialized view is eligible for rewrite, and if so, what rules must be followed:<br>• NONE - Materialized view cannot be used for rewrite, because rewrite is disallowed or prevented<br>• TEXTMATCH - Defining query of the materialized view contained restrictions on the use of query rewrite<br>• GENERAL - Defining query of the materialized view contained no restrictions on the use of query rewrite, so Oracle Database can apply any rewrite rule that is supported |
| REFRESH_MODE | VARCHAR2(6) | | Refresh mode of the materialized view:<br>• DEMAND - Oracle Database refreshes this materialized view whenever an appropriate refresh procedure is called |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • COMMIT - Oracle Database refreshes this materialized view when a transaction on one of the materialized view's masters commits<br>• NEVER - Oracle Database never refreshes this materialized view |
| REFRESH_METHOD | VARCHAR2(8) | | Default method used to refresh the materialized view (can be overridden through the API):<br>• COMPLETE - Materialized view is completely refreshed from the masters<br>• FORCE - Oracle Database performs a fast refresh if possible, otherwise a complete refresh<br>• FAST - Oracle Database performs an incremental refresh applying changes that correspond to changes in the masters since the last refresh<br>• NEVER - User specified that Oracle Database should not refresh this materialized view |
| BUILD_MODE | VARCHAR2(9) | | Indicates how the materialized view was populated during creation:<br>• IMMEDIATE - Populated from the masters during creation<br>• DEFERRED - Not populated during creation. Must be explicitly populated later by the user.<br>• PREBUILT - Populated with an existing table during creation. The relationship of the contents of this prebuilt table to the materialized view's masters is unknown to Oracle Database. |
| FAST_REFRESHABLE | VARCHAR2(18) | | Indicates whether the materialized view is eligible for incremental (fast) refresh. Oracle Database calculates this value statically, based on the materialized view definition query:<br>• NO - Materialized view is not fast refreshable, and hence is complex<br>• DML - Fast refresh is supported only for DML operations<br>• DIRLOAD_DML - Fast refresh is supported for both direct loads and DML operations<br>• DIRLOAD_LIMITEDDML - Fast refresh is supported for direct loads and a subset of DML operations |
| LAST_REFRESH_TYPE | VARCHAR2(8) | | Method used for the most recent refresh:<br>• COMPLETE - Most recent refresh was complete<br>• FAST - Most recent refresh was fast (incremental)<br>• NA - Materialized view has not yet been refreshed (for example, if it was created DEFERRED) |
| LAST_REFRESH_DATE | DATE | | Date on which the materialized view was most recently refreshed (blank if not yet populated) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_REFRESH_END_TIME[1] | DATE | | End time of the most recent refresh on the materialized view (blank if not yet populated) |
| STALENESS | VARCHAR2(19) | | Relationship between the contents of the materialized view and the contents of the materialized view's masters:<br>• FRESH - Materialized view is a read-consistent view of the current state of its masters<br>• STALE - Materialized view is out of date because one or more of its masters has changed. If the materialized view was FRESH before it became STALE, then it is a read-consistent view of a former state of its masters.<br>• NEEDS_COMPILE - Some object upon which the materialized view depends has changed. An ALTER MATERIALIZED VIEW...COMPILE statement is required to validate this materialized view and compute the staleness of the contents.<br>• UNUSABLE - Materialized view is not a read-consistent view of its masters from any point in time<br>• UNKNOWN - Oracle Database does not know whether the materialized view is in a read-consistent view of its masters from any point in time (this is the case for materialized views created on prebuilt tables)<br>• UNDEFINED - Materialized view has remote masters. The concept of staleness is not defined for such materialized views. |
| AFTER_FAST_REFRESH | VARCHAR2(19) | | Specifies the staleness value that will occur if a fast refresh is applied to this materialized view. Its values are the same as for the STALENESS column, plus the value NA, which is used when fast refresh is not applicable to this materialized view. |
| UNKNOWN_PREBUILT | VARCHAR2(1) | | Indicates whether the materialized view is prebuilt (Y) or not (N) |
| UNKNOWN_PLSQL_FUNC | VARCHAR2(1) | | Indicates whether the materialized view contains PL/SQL functions (Y) or not (N) |
| UNKNOWN_EXTERNAL_TABLE | VARCHAR2(1) | | Indicates whether the materialized view contains external tables (Y) or not (N) |
| UNKNOWN_CONSIDER_FRESH | VARCHAR2(1) | | Indicates whether the materialized view is considered fresh (Y) or not (N) |
| UNKNOWN_IMPORT | VARCHAR2(1) | | Indicates whether the materialized view is imported (Y) or not (N) |
| UNKNOWN_TRUSTED_FD | VARCHAR2(1) | | Indicates whether the materialized view uses trusted constraints for refresh (Y) or not (N) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPILE_STATE | VARCHAR2(19) | | Validity of the materialized view with respect to the objects upon which it depends:<br>• VALID - Materialized view has been validated without error, and no object upon which it depends has changed since the last validation<br>• NEEDS_COMPILE - Some object upon which the materialized view depends has changed. An ALTER MATERIALIZED VIEW...COMPILE statement is required to validate this materialized view.<br>• ERROR - Materialized view has been validated with one or more errors |
| USE_NO_INDEX | VARCHAR2(1) | | Indicates whether the materialized view was created using the USING NO INDEX clause (Y) or the materialized view was created with the default index (N). The USING NO INDEX clause suppresses the creation of the default index. |
| STALE_SINCE | DATE | | Time from when the materialized view became stale |
| NUM_PCT_TABLES | NUMBER | | Number of PCT detail tables |
| NUM_FRESH_PCT_REGIONS | NUMBER | | Number of fresh PCT partition regions |
| NUM_STALE_PCT_REGIONS | NUMBER | | Number of stale PCT partition regions |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the materialized view was created using the SEGMENT CREATION DEFERRED clause. The value is YES if the segment for the materialized view is created and NO if it is not. |
| EVALUATION_EDITION | VARCHAR2(128) | | Name of the edition in which editioned objects referenced in an expression column are resolved |
| UNUSABLE_BEFORE | VARCHAR2(128) | | Name of the oldest edition in which the stored results of the materialized view's subquery may be used for query rewrite. In editions before the specified edition, the stored results of the materialized view's data are considered unusable. This value is NULL if no such edition is specified. |
| UNUSABLE_BEGINNING | VARCHAR2(128) | | Name of the oldest edition in which the stored results of the materialized view's subquery may *not* be used for query rewrite. The data is unusable for query rewrite in the specified edition and in any descendants of this edition. This value is NULL if no such edition is specified. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_MVIEWS"
- "USER_MVIEWS"
- *Oracle Database Advanced Replication* for more information on materialized views to support replication
- *Oracle Database Data Warehousing Guide* for more information on materialized views to support data warehousing

# ALL_NESTED_TABLE_COLS

ALL_NESTED_TABLE_COLS describes the columns of the nested tables accessible to the current user. To gather statistics for this view, use the DBMS_STATS package.

### Related Views

- DBA_NESTED_TABLE_COLS describes the columns of all nested tables in the database.
- USER_NESTED_TABLE_COLS describes the columns of the nested tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the nested table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the nested table |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name |
| DATA_TYPE | VARCHAR2(128) | | Data type of the column |
| DATA_TYPE_MOD | VARCHAR2(3) | | Data type modifier of the column |
| DATA_TYPE_OWNER | VARCHAR2(128) | | Owner of the data type of the column |
| DATA_LENGTH | NUMBER | NOT NULL | Length of the column (in bytes) |
| DATA_PRECISION | NUMBER | | Decimal precision for NUMBER data type; binary precision for FLOAT data type; NULL for all other data types |
| DATA_SCALE | NUMBER | | Digits to the right of the decimal point in a number |
| NULLABLE | VARCHAR2(1) | | Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. |
| COLUMN_ID | NUMBER | | Sequence number of the column as created |
| DEFAULT_LENGTH | NUMBER | | Length of the default value for the column |
| DATA_DEFAULT | LONG | | Default value for the column |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column[1] |
| HIGH_VALUE | RAW(1000) | | High value in the column[1] |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/ NUM_DISTINCT.[1] |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| NUM_BUCKETS | NUMBER | | Number of buckets in the histogram for the column<br>**Note:** The number of buckets in a histogram is specified in the SIZE parameter of the SQL ANALYZE statement. However, Oracle Database does not create a histogram with more buckets than the number of rows in the sample. Also, if the sample contains any values that are very repetitious, Oracle Database creates the specified number of buckets, but the value indicated by this column may be smaller because of an internal compression algorithm. |
| LAST_ANALYZED | DATE | | Date on which this column was most recently analyzed |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this column |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Name of the character set:<br>• CHAR_CS<br>• NCHAR_CS |
| CHAR_COL_DECL_LENGTH | NUMBER | | Declaration length of the character type column |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table (YES) or they were not gathered (NO). Note that for partitiioned tables, table level (global statistics) are estimated from partition level statistics when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| CHAR_LENGTH | NUMBER | | Displays the length of the column in characters. This value only applies to the following data types:<br>• CHAR<br>• VARCHAR2 |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NCHAR<br>• NVARCHAR2 |
| CHAR_USED | VARCHAR2(1) | | Indicates that the column uses BYTE length semantics (B) or CHAR length semantics (C), or whether the data type is not any of the following (NULL):<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| V80_FMT_IMAGE | VARCHAR2(3) | | Indicates whether the column data is in release 8.0 image format (YES) or not (NO) |
| DATA_UPGRADED | VARCHAR2(3) | | Indicates whether the column data has been upgraded to the latest type version format (YES) or not (NO) |
| HIDDEN_COLUMN | VARCHAR2(3) | | Indicates whether the column is a hidden column (YES) or not (NO) |
| VIRTUAL_COLUMN | VARCHAR2(3) | | Indicates whether the column is a virtual column (YES) or not (NO) |
| SEGMENT_COLUMN_ID | NUMBER | | Sequence number of the column in the segment |
| INTERNAL_COLUMN_ID | NUMBER | NOT NULL | Internal sequence number of the column |
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE<br>• FREQUENCY<br>• HEIGHT BALANCED |
| QUALIFIED_COL_NAME | VARCHAR2(4000) | | Qualified column name |

[1] These columns remain for backward compatibility with Oracle7. This information is now in the {TAB|PART}_COL_STATISTICS views.

**See Also:**

- "DBA_NESTED_TABLE_COLS"
- "USER_NESTED_TABLE_COLS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_NESTED_TABLES

ALL_NESTED_TABLES describes the nested tables in tables accessible to the current user.

### Related Views

- DBA_NESTED_TABLES describes all nested tables in the database.
- USER_NESTED_TABLES describes nested tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the nested table |
| TABLE_NAME | VARCHAR2(128) | | Name of the nested table |
| TABLE_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of which the nested table was created |
| TABLE_TYPE_NAME | VARCHAR2(128) | | Name of the type of the nested table |
| PARENT_TABLE_NAME | VARCHAR2(128) | | Name of the parent table containing the nested table |
| PARENT_TABLE_COLUMN | VARCHAR2(4000) | | Column name of the parent table that corresponds to the nested table |
| STORAGE_SPEC | VARCHAR2(30) | | Indicates whether storage for the nested table is USER_SPECIFIED or DEFAULT |
| RETURN_TYPE | VARCHAR2(20) | | Return type of the varray column (LOCATOR) or (VALUE) |
| ELEMENT_SUBSTITUTABLE | VARCHAR2(25) | | Indicates whether the nested table element is substitutable (Y) or not (N) |

**See Also:**

- "DBA_NESTED_TABLES"
- "USER_NESTED_TABLES"

# ALL_OBJ_COLATTRS

ALL_OBJ_COLATTRS describes object columns and attributes contained in the tables accessible to the current user.

### Related Views

- DBA_OBJ_COLATTRS describes object columns and attributes contained in all tables in the database.
- USER_OBJ_COLATTRS describes object columns and attributes contained in the tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | | Name of the table containing the object column or attribute |
| COLUMN_NAME | VARCHAR2(4000) | | Fully qualified name of the object column or attribute |
| SUBSTITUTABLE | VARCHAR2(15) | | Indicates whether the column is substitutable (Y) or not (N) |

**See Also:**

- "DBA_OBJ_COLATTRS"
- "USER_OBJ_COLATTRS"

# ALL_OBJECT_TABLES

ALL_OBJECT_TABLES describes the object tables accessible to the current user.

### Related Views

- DBA_OBJECT_TABLES describes all object tables in the database.
- USER_OBJECT_TABLES describes the object tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the table; NULL for partitioned, temporary, and index-organized tables |
| CLUSTER_NAME | VARCHAR2(128) | | Name of the cluster, if any, to which the table belongs |
| IOT_NAME | VARCHAR2(128) | | Name of the index-organized table, if any, to which the overflow or mapping table entry belongs. If the IOT_TYPE column is not NULL, then this column contains the base table name. |
| STATUS | VARCHAR2(8) | | If a previous DROP TABLE operation failed, indicates whether the table is unusable (UNUSABLE) or valid (VALID) |
| PCT_FREE | NUMBER | | Minimum percentage of free space in a block; NULL for partitioned tables |
| PCT_USED | NUMBER | | Minimum percentage of used space in a block; NULL for partitioned tables |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INI_TRANS | NUMBER | | Initial number of transactions; NULL for partitioned tables |
| MAX_TRANS | NUMBER | | Maximum number of transactions; NULL for partitioned tables |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent (in bytes); NULL for partitioned tables |
| NEXT_EXTENT | NUMBER | | Size of secondary extents (in bytes); NULL for partitioned tables |
| MIN_EXTENTS | NUMBER | | Minimum number of extents allowed in the segment; NULL for partitioned tables |
| MAX_EXTENTS | NUMBER | | Maximum number of extents allowed in the segment; NULL for partitioned tables |
| PCT_INCREASE | NUMBER | | Percentage increase in extent size; NULL for partitioned tables |
| FREELISTS | NUMBER | | Number of process freelists allocated to the segment; NULL for partitioned tables |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated to the segment; NULL for partitioned tables |
| LOGGING | VARCHAR2(3) | | Indicates whether or not changes to the table are logged:<br>• YES<br>• NO |
| BACKED_UP | VARCHAR2(1) | | Indicates whether the table has been backed up since the last modification (Y) or not (N) |
| NUM_ROWS | NUMBER | | Number of rows in the table |
| BLOCKS | NUMBER | | Number of used blocks in the table |
| EMPTY_BLOCKS | NUMBER | | Number of empty (never used) blocks in the table |
| AVG_SPACE | NUMBER | | Average available free space in the table |
| CHAIN_CNT | NUMBER | | Number of chained rows in the table |
| AVG_ROW_LEN | NUMBER | | Average row length, including row overhead |
| AVG_SPACE_FREELIST_BLOCKS | NUMBER | | Average free space of all blocks on a freelist |
| NUM_FREELIST_BLOCKS | NUMBER | | Number of blocks on the freelist |
| DEGREE | VARCHAR2(10) | | Number of parallel execution processes per instance for scanning the table, or DEFAULT |
| INSTANCES | VARCHAR2(10) | | Number of instances across which the table is to be scanned, or DEFAULT |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CACHE | VARCHAR2(5) | | Indicates whether the table is to be cached in the buffer cache (Y) or not (N) |
| TABLE_LOCK | VARCHAR2(8) | | Indicates whether table locking is enabled (ENABLED) or disabled (DISABLED) |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this table |
| LAST_ANALYZED | DATE | | Date on which this table was most recently analyzed |
| PARTITIONED | VARCHAR2(3) | | Indicates whether the table is partitioned (YES) or not (NO) |
| IOT_TYPE | VARCHAR2(12) | | If the table is an index-organized table, then IOT_TYPE is IOT, IOT_OVERFLOW, or IOT_MAPPING. If the table is not an index-organized table, then IOT_TYPE is NULL. |
| OBJECT_ID_TYPE | VARCHAR2(16) | | Indicates whether the object ID (OID) is USER-DEFINED or SYSTEM GENERATED |
| TABLE_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the table |
| TABLE_TYPE | VARCHAR2(128) | | Type of the table |
| TEMPORARY | VARCHAR2(1) | | Indicates whether this is a temporary table (Y) or not (N) |
| SECONDARY | VARCHAR2(1) | | Indicates whether the object table is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N) |
| NESTED | VARCHAR2(3) | | Indicates whether the table is a nested table (YES) or not (NO) |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool to be used for table blocks:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for table blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for table blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| ROW_MOVEMENT | VARCHAR2(8) | | Indicates whether partitioned row movement is enabled (ENABLED) or disabled (DISABLED) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the object as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned objects, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| DURATION | VARCHAR2(15) | | Indicates the duration of a temporary table:<br>• SYS$SESSION - Rows are preserved for the duration of the session<br>• SYS$TRANSACTION - Rows are deleted after COMMIT<br>Null - Permanent table |
| SKIP_CORRUPT | VARCHAR2(8) | | Indicates whether Oracle Database ignores blocks marked corrupt during table and index scans (ENABLED) or raises an error (DISABLED). To enable this feature, run the DBMS_REPAIR.SKIP_CORRUPT_BLOCKS procedure. |
| MONITORING | VARCHAR2(3) | | Indicates whether the table has the MONITORING attribute set (YES) or not (NO) |
| CLUSTER_OWNER | VARCHAR2(128) | | Owner of the cluster, if any, to which the table belongs |
| DEPENDENCIES | VARCHAR2(8) | | Indicates whether row-level dependency tracking is enabled (ENABLED) or disabled (DISABLED) |
| COMPRESSION | VARCHAR2(8) | | Indicates whether table compression is enabled (ENABLED) or not (DISABLED); NULL for partitioned tables |
| COMPRESS_FOR | VARCHAR2(30) | | Default compression for what kind of operations:<br>• BASIC<br>• ADVANCED<br>• QUERY LOW<br>• QUERY HIGH[1]<br>• ARCHIVE LOW[1]<br>• ARCHIVE HIGH[1]<br>• NULL |
| DROPPED | VARCHAR2(3) | | Indicates whether the table has been dropped and is in the recycle bin (YES) or not (NO); NULL for partitioned tables |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the table segment has been created (YES) or not (NO) |
| INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is enabled (ENABLED) or disabled (DISABLED) for this segment |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INMEMORY_PRIORITY[2] | VARCHAR2(8) | | Indicates the priority for In-Memory Column Store (IM column store) population. Possible values:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| INMEMORY_DISTRIBUTE[2] | VARCHAR2(15) | | Indicates how the IM column store is distributed in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| INMEMORY_COMPRESSION[2] | VARCHAR2(17) | | Indicates the compression level for the IM column store:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ]<br>• NULL[3] |
| INMEMORY_DUPLICATE[2] | VARCHAR2(13) | | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |

1 Hybrid Columnar Compression is a feature of the Enterprise Edition of Oracle Database. See *Oracle Database Concepts* for more information.

2 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

3 This column has a value based on where the segments lie for a table. For example, if the table is partitioned and is enabled for the IM column store, the value is NULL for ALL_TABLES but non-NULL for ALL_TAB_PARTITIONS.

**See Also:**

- "DBA_OBJECT_TABLES"
- "USER_OBJECT_TABLES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_REPAIR.SKIP_CORRUPT_BLOCKS procedure

# ALL_OBJECTS

ALL_OBJECTS describes all objects accessible to the current user.

### Related Views

- DBA_OBJECTS describes all objects in the database.
- USER_OBJECTS describes all objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject (for example, partition) |
| OBJECT_ID | NUMBER | NOT NULL | Dictionary object number of the object |
| DATA_OBJECT_ID | NUMBER | | Dictionary object number of the segment that contains the object.<br>**Note:** OBJECT_ID and DATA_OBJECT_ID display data dictionary metadata. Do not confuse these numbers with the unique 16-byte object identifier (*object ID*) that Oracle Database assigns to row objects in object tables in the system. |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object (such as TABLE, INDEX) |
| CREATED | DATE | NOT NULL | Timestamp for the creation of the object |
| LAST_DDL_TIME | DATE | NOT NULL | Timestamp for the last modification of the object and dependent objects resulting from a DDL statement (including grants and revokes) |
| TIMESTAMP | VARCHAR2(19) | | Timestamp for the specification of the object (character data) |
| STATUS | VARCHAR2(7) | | Status of the object:<br>• VALID<br>• INVALID<br>• N/A |
| TEMPORARY | VARCHAR2(1) | | Indicates whether the object is temporary (the current session can see only data that it placed in this object itself) (Y) or not (N) |
| GENERATED | VARCHAR2(1) | | Indicates whether the name of this object was system-generated (Y) or not (N) |
| SECONDARY | VARCHAR2(1) | | Indicates whether this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N) |
| NAMESPACE | NUMBER | NOT NULL | Namespace for the object |
| EDITION_NAME | VARCHAR2(128) | | Name of the edition in which the object is actual |
| SHARING | VARCHAR2(13) | | Values:<br>• METADATA LINK - If the object is metadata-linked or a metadata link to an object in the root |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • OBJECT LINK - If the object is object-linked or an object link to an object in the root<br>• NONE - If none of the above applies |
| EDITIONABLE | VARCHAR2(1) | | Values:<br>• Y - For objects marked EDITIONABLE<br>• N - For objects marked NONEDITIONABLE<br>• NULL - For objects whose type is not editionable in the database |
| ORACLE_MAINTAINED | VARCHAR2(1) | | Denotes whether the object was created, and is maintained, by Oracle-supplied scripts (such as catalog.sql or catproc.sql). An object for which this column has the value Y must not be changed in any way except by running an Oracle-supplied script. |

**See Also:**

- "DBA_OBJECTS"
- "USER_OBJECTS"

# ALL_OBJECTS_AE

ALL_OBJECTS_AE describes the objects (across all editions) accessible to the current user. Dropped objects appear in this view with OBJECT_TYPE = NON-EXISTENT.

### Related Views

- DBA_OBJECTS_AE describes all objects (across all editions) in the database.
- USER_OBJECTS_AE describes the objects (across all editions) owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject (for example, partition) |
| OBJECT_ID | NUMBER | NOT NULL | Dictionary object number of the object |
| DATA_OBJECT_ID | NUMBER | | Dictionary object number of the segment which contains the object |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object |
| CREATED | DATE | NOT NULL | Timestamp for the creation of the object |
| LAST_DDL_TIME | DATE | NOT NULL | Timestamp for the last modification of the object and dependent objects resulting from a DDL statement (including grants and revokes) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TIMESTAMP | VARCHAR2(19) | | Timestamp for the specification of the object (character data) |
| STATUS | VARCHAR2(7) | | Status of the object:<br>• VALID<br>• INVALID<br>• N/A |
| TEMPORARY | VARCHAR2(1) | | Indicates whether the object is temporary (the current session can see only data that it placed in this object itself) (Y) or not (N) |
| GENERATED | VARCHAR2(1) | | Indicates whether the name of this object was system-generated (Y) or not (N) |
| SECONDARY | VARCHAR2(1) | | Indicates whether this is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N) |
| NAMESPACE | NUMBER | NOT NULL | Namespace for the object |
| EDITION_NAME | VARCHAR2(128) | | Name of the edition in which the object is actual |
| EDITIONABLE | VARCHAR2(1) | | Values:<br>• Y - For objects marked EDITIONABLE<br>• N - For objects marked NONEDITIONABLE<br>• NULL - For objects whose type is not editionable in the database |
| ORACLE_MAINTAINED | VARCHAR2(1) | | Denotes whether the object was created, and is maintained, by Oracle-supplied scripts (such as catalog.sql or catproc.sql). An object for which this column has the value Y must not be changed in any way except by running an Oracle-supplied script. |

**See Also:**

- "DBA_OBJECTS_AE"
- "USER_OBJECTS_AE"

# ALL_OPANCILLARY

ALL_OPANCILLARY describes operators whose bindings are ancillary to other (primary) operators.

### Related Views

- DBA_OPANCILLARY describes such information about all operators in the database.
- USER_OPANCILLARY describes such information about operators owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the ancillary operator |
| OPERATOR_NAME | VARCHAR2(128) | NOT NULL | Name of the ancillary operator |
| BINDING# | NUMBER | NOT NULL | Binding number of the ancillary operator |
| PRIMOP_OWNER | VARCHAR2(128) | NOT NULL | Owner of the primary operator |
| PRIMOP_NAME | VARCHAR2(128) | NOT NULL | Name of the primary operator |
| PRIMOP_BIND# | NUMBER | NOT NULL | Binding number of the primary operator |

**See Also:**

- "DBA_OPANCILLARY"
- "USER_OPANCILLARY"

# ALL_OPARGUMENTS

ALL_OPARGUMENTS describes arguments for each operator binding accessible to the current user.

### Related Views

- DBA_OPARGUMENTS describes arguments of all operator bindings in the database.
- USER_OPARGUMENTS describes arguments of all operator bindings owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the operator argument |
| OPERATOR_NAME | VARCHAR2(128) | NOT NULL | Name of the operator argument |
| BINDING# | NUMBER | NOT NULL | Binding number of the operator argument |
| POSITION | NUMBER | NOT NULL | Position of the operator argument (1, 2, 3, ...) |
| ARGUMENT_TYPE | VARCHAR2(61) | | Datatype of the operator argument |

**See Also:**

- "DBA_OPARGUMENTS"
- "USER_OPARGUMENTS"

# ALL_OPBINDINGS

ALL_OPBINDINGS describes the binding functions and methods on the operators accessible to the current user.

### Related Views

- DBA_OPBINDINGS describes the binding functions and methods on all operators in the database.
- USER_OPBINDINGS describes the binding functions and methods on the operators owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the operator |
| OPERATOR_NAME | VARCHAR2(128) | NOT NULL | Name of the operator |
| BINDING# | NUMBER | NOT NULL | Binding number of the operator |
| FUNCTION_NAME | VARCHAR2(92) | | Name of the binding function or method as specified by the user |
| RETURN_SCHEMA | VARCHAR2(128) | | Name of the schema of the return type if the return type of the binding is an object type |
| RETURN_TYPE | VARCHAR2(128) | | Name of the return type |
| IMPLEMENTATION_TYPE_SCHEMA | VARCHAR2(128) | | If the operator was created WITH INDEX CONTEXT or SCAN CONTEXT, then this column displays the schema of the implementation type used by the functional implementation of the operator as a scan context (null if the operator was created without this syntax).<br>**See Also:** the CREATE OPERATOR statement in *Oracle Database SQL Language Reference* |
| IMPLEMENTATION_TYPE | VARCHAR2(128) | | If the operator was created WITH INDEX CONTEXT or SCAN CONTEXT, then this column displays the name of the implementation type used by the functional implementation of the operator as a scan context (null if the operator was created without this syntax).<br>**See Also:** the CREATE OPERATOR statement in *Oracle Database SQL Language Reference* |
| PROPERTY | VARCHAR2(43) | | Property of the operator binding:<br>• WITH INDEX CONTEXT<br>• COMPUTE ANCILLARY DATA<br>• ANCILLARY TO<br>• WITH COLUMN CONTEXT<br>• WITH INDEX, COLUMN CONTEXT<br>• COMPUTE ANCILLARY DATA, WITH COLUMN CONTEXT |

**See Also:**

- "DBA_OPBINDINGS"
- "USER_OPBINDINGS"

# ALL_OPERATOR_COMMENTS

ALL_OPERATOR_COMMENTS displays comments for the user-defined operators accessible to the current user.

### Related Views

- DBA_OPERATOR_COMMENTS displays comments for all user-defined operators in the database.
- USER_OPERATOR_COMMENTS displays comments for the user-defined operators owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the user-defined operator |
| OPERATOR_NAME | VARCHAR2(128) | NOT NULL | Name of the user-defined operator |
| COMMENTS | VARCHAR2(4000) | | Comment for the user-defined operator |

**See Also:**

- "DBA_OPERATOR_COMMENTS"
- "USER_OPERATOR_COMMENTS"

# ALL_OPERATORS

ALL_OPERATORS describes the operators accessible to the current user.

### Related Views

- DBA_OPERATORS describes all operators in the database.
- USER_OPERATORS describes the operators owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the operator |
| OPERATOR_NAME | VARCHAR2(128) | NOT NULL | Name of the operator |
| NUMBER_OF_BINDS | NUMBER | NOT NULL | Number of bindings associated with the operator |

**See Also:**

- "DBA_OPERATORS"
- "USER_OPERATORS"

# ALL_OUTLINE_HINTS

ALL_OUTLINE_HINTS is a synonym for USER_OUTLINE_HINTS.

**See Also:**

"USER_OUTLINE_HINTS"

# ALL_OUTLINES

ALL_OUTLINES is a synonym for USER_OUTLINES.

**See Also:**

"USER_OUTLINES"

# 3

# Static Data Dictionary Views: ALL_PART_COL_STATISTICS to DATABASE_PROPERTIES

This chapter contains the static data dictionary views ALL_PART_COL_STATISTICS through DATABASE_PROPERTIES.

## ALL_PART_COL_STATISTICS

ALL_PART_COL_STATISTICS displays column statistics and histogram information for the table partitions accessible to the current user.

### Related Views

- DBA_PART_COL_STATISTICS displays column statistics and histogram information for all table partitions in the database.
- USER_PART_COL_STATISTICS displays column statistics and histogram information for the table partitions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the partitioned table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the table partition |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column |
| HIGH_VALUE | RAW(1000) | | High value in the column |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/NUM_DISTINCT. |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NUM_BUCKETS | NUMBER | | Number of buckets in histogram for the column |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing the column |
| LAST_ANALYZED | DATE | | Date on which the column was most recently analyzed |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the partition as a whole (YES) or they were not gathered (NO). Note that partition statistics are estimated from subpartition level statistics when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| NOTES | VARCHAR2(41) | | Describes some additional properties of the statistics. For example, if the value is INCREMENTAL, the global statistics are derived from synopses, that is, the global statistics are incrementally maintained. |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE<br>• FREQUENCY<br>• HEIGHT BALANCED<br>• HYBRID<br>• TOP-FREQUENCY |

**See Also:**

- "DBA_PART_COL_STATISTICS"
- "USER_PART_COL_STATISTICS"

# ALL_PART_HISTOGRAMS

ALL_PART_HISTOGRAMS displays the histogram data (endpoints per histogram) for the histograms on the table partitions accessible to the current user.

### Related Views

- DBA_PART_HISTOGRAMS displays the histogram data for the histograms on all table partitions in the database.
- USER_PART_HISTOGRAMS displays the histogram data for the histograms on the table partitions owned by the current user. This view does not display the OWNER column.

**Note:**

These views are populated only if you collect statistics on the index using the DBMS_STATS package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the table partition |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column |
| BUCKET_NUMBER | NUMBER | | Bucket number of the histogram |
| ENDPOINT_VALUE | NUMBER | | Normalized endpoint values for the bucket |
| ENDPOINT_ACTUAL_VALUE | VARCHAR2(4000) | | Actual (not normalized) string value of the endpoint for the bucket |
| ENDPOINT_ACTUAL_VALUE_RAW | RAW(1000) | | Endpoint actual value in raw format |
| ENDPOINT_REPEAT_COUNT | NUMBER | | Frequency of the endpoint (applies only to hybrid histograms, and is set to 0 for other histogram types) |

**See Also:**

- "DBA_PART_HISTOGRAMS"
- "USER_PART_HISTOGRAMS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_PART_INDEXES

ALL_PART_INDEXES displays the object-level partitioning information for the partitioned indexes accessible to the current user.

### Related Views

- DBA_PART_INDEXES displays the object-level partitioning information for all partitioned indexes in the database.
- USER_PART_INDEXES displays the object-level partitioning information for the partitioned indexes owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the partitioned index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the partitioned index |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the partitioned table |
| PARTITIONING_TYPE | VARCHAR2(9) | | Type of the partitioning method:<br>• NONE - Not specified<br>**See Also:** the *_INDEXES view<br>• RANGE<br>• HASH<br>• SYSTEM<br>• LIST<br>• REFERENCE |
| SUBPARTITIONING_TYPE | VARCHAR2(9) | | Type of the composite partitioning method:<br>• NONE - Not specified<br>**See Also:** the *_INDEXES view<br>• RANGE<br>• HASH<br>• SYSTEM<br>• LIST<br>• REFERENCE |
| PARTITION_COUNT | NUMBER | NOT NULL | Number of partitions in the index |
| DEF_SUBPARTITION_COUNT | NUMBER | | For a composite-partitioned index, the default number of subpartitions, if specified |
| PARTITIONING_KEY_COUNT | NUMBER | NOT NULL | Number of columns in the partitioning key |
| SUBPARTITIONING_KEY_COUNT | NUMBER | | For a composite-partitioned index, the number of columns in the subpartitioning key |
| LOCALITY | VARCHAR2(6) | | Indicates whether the partitioned index is local (LOCAL) or global (GLOBAL) |
| ALIGNMENT | VARCHAR2(12) | | Indicates whether the partitioned index is prefixed (PREFIXED) or non-prefixed (NON_PREFIXED) |
| DEF_TABLESPACE_NAME | VARCHAR2(30) | | For a local index, the default tablespace to be used when adding or splitting a table partition |
| DEF_PCT_FREE | NUMBER | NOT NULL | For a local index, the default PCTFREE value to be used when adding a table partition |
| DEF_INI_TRANS | NUMBER | NOT NULL | For a local index, the default INITRANS value to be used when adding a table partition |
| DEF_MAX_TRANS | NUMBER | NOT NULL | For a local index, the default MAXTRANS value to be used when adding a table partition |
| DEF_INITIAL_EXTENT | VARCHAR2(40) | | For a local index, the default INITIAL value (in Oracle blocks) to be used when adding a table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | partition, or DEFAULT if no INITIAL value was specified |
| DEF_NEXT_EXTENT | VARCHAR2(40) | | For a local index, the default NEXT value (in Oracle blocks) to be used when adding a table partition, or DEFAULT if no NEXT value was specified |
| DEF_MIN_EXTENTS | VARCHAR2(40) | | For a local index, the default MINEXTENTS value to be used when adding a table partition, or DEFAULT if no MINEXTENTS value was specified |
| DEF_MAX_EXTENTS | VARCHAR2(40) | | For a local index, the default MAXEXTENTS value to be used when adding a table partition, or DEFAULT if no MAXEXTENTS value was specified |
| DEF_MAX_SIZE | VARCHAR2(40) | | For a local index, the default MAXSIZE value to be used when adding a table partition, or DEFAULT if no MAXSIZE value was specified |
| DEF_PCT_INCREASE | VARCHAR2(40) | | For a local index, the default PCTINCREASE value to be used when adding a table partition, or DEFAULT if no PCTINCREASE value was specified |
| DEF_FREELISTS | NUMBER | NOT NULL | For a local index, the default FREELISTS value to be used when adding a table partition |
| DEF_FREELIST_GROUPS | NUMBER | NOT NULL | For a local index, the default FREELIST GROUPS value to be used when adding a table partition |
| DEF_LOGGING | VARCHAR2(7) | | For a local index, the default LOGGING attribute to be used when adding a table partition:<br>• NONE - Not specified<br>**See Also:** the *_INDEXES view<br>• YES<br>• NO |
| DEF_BUFFER_POOL | VARCHAR2(7) | | For a local index, the default buffer pool to be used when adding a table partition:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| DEF_FLASH_CACHE | VARCHAR2(7) | | For a local index, the default Database Smart Flash Cache hint to be used when adding a table partition:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| DEF_CELL_FLASH_CACHE | VARCHAR2(7) | | For a local index, the default cell flash cache hint to be used when adding a table partition:<br>• DEFAULT<br>• KEEP |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| DEF_PARAMETERS | VARCHAR2(1000) | | Default parameter string for domain indexes |
| INTERVAL | VARCHAR2(1000) | | String of the interval value |

**See Also:**

- "DBA_PART_INDEXES"
- "USER_PART_INDEXES"

# ALL_PART_KEY_COLUMNS

ALL_PART_KEY_COLUMNS describes the partitioning key columns for the partitioned objects accessible to the current user.

### Related Views

- DBA_PART_KEY_COLUMNS describes the partitioning key columns for all partitioned objects in the database.
- USER_PART_KEY_COLUMNS describes the partitioning key columns for the partitioned objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the partitioned table or index |
| NAME | VARCHAR2(128) | | Name of the partitioned table or index |
| OBJECT_TYPE | CHAR(5) | | Object type:<br>• TABLE<br>• INDEX |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column |
| COLUMN_POSITION | NUMBER | | Position of the column within the partitioning key |

**See Also:**

- "DBA_PART_KEY_COLUMNS"
- "USER_PART_KEY_COLUMNS"

# ALL_PART_LOBS

ALL_PART_LOBS displays table-level information about the partitioned LOBs accessible to the current user, including default attributes for LOB data partitions.

### Related Views

- DBA_PART_LOBS displays table-level information about all partitioned LOBs in the database.
- USER_PART_LOBS displays table-level information about the partitioned LOBs owned by the current user. This view does not display the TABLE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the partitioned table containing the LOBs |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the partitioned table containing the LOBs |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the LOB column |
| LOB_NAME | VARCHAR2(128) | NOT NULL | Name of the partitioned LOB |
| LOB_INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the partitioned LOB index |
| DEF_CHUNK | NUMBER | NOT NULL | Default value of CHUNK for a LOB data partition to be used when adding a partition |
| DEF_PCTVERSION | NUMBER | NOT NULL | Default value of PCTVERSION for a LOB data partition to be used when adding a partition |
| DEF_CACHE | VARCHAR2(10) | | Indicates whether and how the LOB data is cached by default in the buffer cache:<br>• YES - LOB data is placed in the buffer cache<br>• NO - LOB data either is not brought into the buffer cache or is brought into the buffer cache and placed at the least recently used end of the LRU list<br>• CACHEREADS - LOB data is brought into the buffer cache only during read operations but not during write operations |
| DEF_IN_ROW | VARCHAR2(3) | | Indicates whether LOB data < 4000 bytes is stored by default inline (in the row) (YES) or not (NO); that is, whether or not ENABLE STORAGE IN ROW was specified when the LOB column was created or last altered |
| DEF_TABLESPACE_NAME | VARCHAR2(30) | | Default tablespace for a LOB data partition to be used when adding a partition |
| DEF_INITIAL_EXTENT | VARCHAR2(40) | | Default value of INITIAL for a LOB data partition to be used when adding a partition, or DEFAULT if no INITIAL value was specified |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DEF_NEXT_EXTENT | VARCHAR2(40) | | Default value of NEXT for a LOB data partition to be used when adding a partition, or DEFAULT if no NEXT value was specified |
| DEF_MIN_EXTENTS | VARCHAR2(40) | | Default value of MINEXTENTS for a LOB data partition to be used when adding a partition, or DEFAULT if no MINEXTENTS value was specified |
| DEF_MAX_EXTENTS | VARCHAR2(40) | | Default value of MAXEXTENTS for a LOB data partition to be used when adding a partition, or DEFAULT if no MAXEXTENTS value was specified |
| DEF_MAX_SIZE | VARCHAR2(40) | | Default value of MAXSIZE for a LOB data partition to be used when adding a partition, or DEFAULT if no MAXSIZE value was specified |
| DEF_RETENTION | VARCHAR2(7) | | Default value of RETENTION for a LOB data partition to be used when adding a partition.<br>Possible values for SecureFiles:<br>• NONE<br>• AUTO<br>• MIN<br>• MAX<br>• DEFAULT<br>• INVALID<br>Possible values for BasicFiles:<br>• YES<br>• NO |
| DEF_MINRET | VARCHAR2(40) | | Default value of RETENTION MIN for a LOB data partition to be used when adding a partition, or DEFAULT if no RETENTION MIN value was specified |
| DEF_PCT_INCREASE | VARCHAR2(40) | | Default value of PCTINCREASE for a LOB data partition to be used when adding a partition, or DEFAULT if no PCTINCREASE value was specified |
| DEF_FREELISTS | VARCHAR2(40) | | Default value of FREELISTS for a LOB data partition to be used when adding a partition, or DEFAULT if no FREELISTS value was specified |
| DEF_FREELIST_GROUPS | VARCHAR2(40) | | Default value of FREELIST GROUPS for a LOB data partition to be used when adding a partition, or DEFAULT if no FREELIST GROUPS value was specified |
| DEF_LOGGING | VARCHAR2(7) | | Default LOGGING attribute for a LOB data partition to be used when adding a partition:<br>• NONE - Not specified<br>**See Also:** the *_LOBS and *_LOB_PARTITIONS views<br>• YES<br>• NO |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DEF_BUFFER_POOL | VARCHAR2(7) | | Default buffer pool for a LOB data partition to be used when adding a partition:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| DEF_FLASH_CACHE | VARCHAR2(7) | | Default Database Smart Flash Cache hint to be used when adding a partition:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| DEF_CELL_FLASH_CACHE | VARCHAR2(7) | | Default cell flash cache hint to be used when adding a partition:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| DEF_ENCRYPT | VARCHAR2(4) | | Default value of ENCRYPT for a LOB data partition to be used when adding a partition.<br>Possible values for SecureFiles:<br>• YES<br>• NO<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| DEF_COMPRESS | VARCHAR2(6) | | Default value of COMPRESS for a LOB data partition to be used when adding a partition.<br>Possible values for SecureFiles:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• NO - Compression is off<br>Possible value for BasicFiles:<br>• NONE - Not applicable |
| DEF_DEDUPLICATE | VARCHAR2(15) | | Default value of DEDUPLICATE for a LOB data partition to be used when adding a partition.<br>Possible values for SecureFiles:<br>• LOB - Deduplicate<br>• NO - Keep duplicates<br>Possible values for BasicFiles:<br>• NONE - Not applicable |
| DEF_SECUREFILE | VARCHAR2(3) | | Indicates whether the LOB is SecureFiles (YES) or not (NO) |

**See Also:**

- "DBA_PART_LOBS"
- "USER_PART_LOBS"

# ALL_PART_TABLES

ALL_PART_TABLES displays the object-level partitioning information for the partitioned tables accessible to the current user.

### Related Views

- DBA_PART_TABLES displays the object-level partitioning information for all partitioned tables in the database.
- USER_PART_TABLES displays the object-level partitioning information for the partitioned tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the partitioned table |
| TABLE_NAME | VARCHAR2(128) | | Name of the partitioned table |
| PARTITIONING_TYPE | VARCHAR2(9) | | Type of the partitioning method:<br>• UNKNOWN - Not specified<br>**See Also:** the *_TABLES view<br>• RANGE<br>• HASH<br>• SYSTEM<br>• LIST<br>• REFERENCE |
| SUBPARTITIONING_TYPE | VARCHAR2(9) | | Type of the composite partitioning method:<br>• NONE - Not subpartitioned<br>**See Also:** the *_TABLES view<br>• RANGE<br>• HASH<br>• SYSTEM<br>• LIST<br>• REFERENCE |
| PARTITION_COUNT | NUMBER | | Number of partitions in the table. For interval partitioned tables, the value of this column is always 1048575. |
| DEF_SUBPARTITION_COUNT | NUMBER | | For a composite-partitioned table, the default number of subpartitions, if specified |
| PARTITIONING_KEY_COUNT | NUMBER | | Number of columns in the partitioning key |
| SUBPARTITIONING_KEY_COUNT | NUMBER | | For a composite-partitioned table, the number of columns in the subpartitioning key |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATUS | VARCHAR2(8) | | If a previous DROP TABLE operation failed, indicates whether the table is unusable (UNUSABLE) or valid (VALID) |
| DEF_TABLESPACE_NAME | VARCHAR2(30) | | Default tablespace to be used when adding a partition |
| DEF_PCT_FREE | NUMBER | | Default value of PCTFREE to be used when adding a partition |
| DEF_PCT_USED | NUMBER | | Default value of PCTUSED to be used when adding a partition |
| DEF_INI_TRANS | NUMBER | | Default value of INITRANS to be used when adding a partition |
| DEF_MAX_TRANS | NUMBER | | Default value of MAXTRANS to be used when adding a partition |
| DEF_INITIAL_EXTENT | VARCHAR2(40) | | Default value of INITIAL (in Oracle blocks) to be used when adding a partition, or DEFAULT if no INITIAL value was specified |
| DEF_NEXT_EXTENT | VARCHAR2(40) | | Default value of NEXT (in Oracle blocks) to be used when adding a partition, or DEFAULT if no NEXT value was specified |
| DEF_MIN_EXTENTS | VARCHAR2(40) | | Default value of MINEXTENTS to be used when adding a partition, or DEFAULT if no MINEXTENTS value was specified |
| DEF_MAX_EXTENTS | VARCHAR2(40) | | Default value of MAXEXTENTS to be used when adding a partition, or DEFAULT if no MAXEXTENTS value was specified |
| DEF_MAX_SIZE | VARCHAR2(40) | | Default value of MAXSIZE to be used when adding a partition, or DEFAULT if no MAXSIZE value was specified |
| DEF_PCT_INCREASE | VARCHAR2(40) | | Default value of PCTINCREASE to be used when adding a partition, or DEFAULT if no PCTINCREASE value was specified |
| DEF_FREELISTS | NUMBER | | Default value of FREELISTS to be used when adding a partition |
| DEF_FREELIST_GROUPS | NUMBER | | Default value of FREELIST GROUPS to be used when adding a partition |
| DEF_LOGGING | VARCHAR2(7) | | Default LOGGING attribute to be used when adding a partition:<br>• NONE - Not specified<br>**See Also:** the *_TABLES view<br>• YES<br>• NO |
| DEF_COMPRESSION | VARCHAR2(8) | | Default compression to be used when adding a partition:<br>• NONE - Not specified |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | **See Also:** the *_TABLES view<br>• ENABLED<br>• DISABLED |
| DEF_COMPRESS_FOR | VARCHAR2(30) | | Default compression for what kind of operations to be used when adding a partition:<br>• BASIC<br>• ADVANCED<br>• QUERY LOW<br>• QUERY HIGH[1]<br>• ARCHIVE LOW[1]<br>• ARCHIVE HIGH[1]<br>• UNKNOWN<br>• NULL |
| DEF_BUFFER_POOL | VARCHAR2(7) | | Default buffer pool to be used when adding a partition:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| DEF_FLASH_CACHE | VARCHAR2(7) | | Default Database Smart Flash Cache hint to be used when adding a partition:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| DEF_CELL_FLASH_CACHE | VARCHAR2(7) | | Default cell flash cache hint to be used when adding a partition:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| REF_PTN_CONSTRAINT_NAME | VARCHAR2(128) | | Name of the partitioning referential constraint for reference-partitioned tables |
| INTERVAL | VARCHAR2(1000) | | String of the interval value |
| IS_NESTED | VARCHAR2(3) | | Indicates whether the partitioned table is a nested table (YES) or not (NO)<br>**See Also:** the *_NESTED_TABLES view for the parent table name/column |
| DEF_SEGMENT_CREATION | VARCHAR2(4) | | Specifies whether the default for segment creation was specified on the table level:<br>• NO - deferred was specified<br>• YES - immediate was specified<br>• NONE - a default for segment creation was not specified |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DEF_INDEXING | VARCHAR2(3) | | Indicates the indexing property specified for the table.<br>Possible values:<br>• ON - INDEXING on was specified explicitly, or no indexing property was specified<br>• OFF - INDEXING off was specified. |
| DEF_INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is by default enabled (ENABLED), disabled (DISABLED), or not specified (NONE) for partitions in this table |
| DEF_INMEMORY_PRIORITY[2] | VARCHAR2(8) | | Indicates the default priority for In-Memory Column Store (IM column store) population. Possible values:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| DEF_INMEMORY_DISTRIBUTE[2] | VARCHAR2(15) | | Indicates how the IM column store is distributed by default for partitions of the table in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| DEF_INMEMORY_COMPRESSION[2] | VARCHAR2(17) | | Default compression level for the IM column store:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ]<br>• NULL |
| DEF_INMEMORY_DUPLICATE[2] | VARCHAR2(13) | | Indicates the default duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |

[1] Hybrid Columnar Compression is a feature of the Enterprise Edition of Oracle Database that is dependent on the underlying storage system. See *Oracle Database Concepts* for more information.
[2] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_PART_TABLES"
- "USER_PART_TABLES"

# ALL_PARTIAL_DROP_TABS

ALL_PARTIAL_DROP_TABS describes tables accessible to the current user that have partially completed DROP COLUMN operations. Such operations might have been interrupted by the user or by a system crash.

### Related Views

- DBA_PARTIAL_DROP_TABS describes all tables in the database that have partially completed DROP COLUMN operations.
- USER_PARTIAL_DROP_TABS describes tables in the schema of the current user that have partially completed DROP COLUMN operations. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |

**See Also:**

- "DBA_PARTIAL_DROP_TABS"
- "USER_PARTIAL_DROP_TABS"

# ALL_PENDING_CONV_TABLES

ALL_PENDING_CONV_TABLES describes the pending conversion tables (tables which are not upgraded to the latest type version) accessible to the current user.

### Related Views

- DBA_PENDING_CONV_TABLES describes all pending conversion tables in the database.
- USER_PENDING_CONV_TABLES describes the pending conversion tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |

**See Also:**

- "DBA_PENDING_CONV_TABLES"
- "USER_PENDING_CONV_TABLES"

# ALL_PLSQL_COLL_TYPES

ALL_PLSQL_COLL_TYPES describes named PL/SQL collection types accessible to the user.

### Related Views

- DBA_PLSQL_COLL_TYPES describes all named PL/SQL collection types in the database. This view does not display the CHAR_USED column.
- USER_PLSQL_COLL_TYPES describes the user's own named PL/SQL collection types. This view does not display the OWNER or CHAR_USED columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | | Name of the type |
| PACKAGE_NAME | VARCHAR2(128) | | Name of the package containing the collection |
| COLL_TYPE | VARCHAR2(128) | | Collection type |
| UPPER_BOUND | NUMBER | | The upper bound of a varray or length constraint of an index by VARCHAR2 table |
| ELEM_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the element |
| ELEM_TYPE_NAME | VARCHAR2(136) | | Name of the type of the element |
| ELEM_TYPE_PACKAGE | VARCHAR2(128) | | Name of the package containing the element |
| LENGTH | NUMBER | | Length of the CHAR element or maximum length of the VARCHAR or VARCHAR2 element |
| PRECISION | NUMBER | | Decimal precision of the NUMBER or DECIMAL element or binary precision of the FLOAT element |
| SCALE | NUMBER | | Scale of the NUMBER or DECIMAL element |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Character set name of the element |
| ELEM_STORAGE | VARCHAR2(7) | | Storage optimization specification for VARRAY of numeric elements |
| NULLS_STORED | VARCHAR2(3) | | Indicates whether null information is stored with each VARRAY element (YES) or not (NO) |
| CHAR_USED | VARCHAR2(1) | | C if the width was specified in characters, B if in bytes |
| INDEX_BY | VARCHAR2(14) | | Index by BINARY_INTEGER or VARCHAR2 |
| ELEM_TYPE_MOD | VARCHAR2(7) | | Type modifier of the element |

**See Also:**

- "DBA_PLSQL_COLL_TYPES"
- "USER_PLSQL_COLL_TYPES"

# ALL_PLSQL_OBJECT_SETTINGS

ALL_PLSQL_OBJECT_SETTINGS displays information about the compiler settings for the stored objects accessible to the current user.

### Related Views

- DBA_PLSQL_OBJECT_SETTINGS displays information about the compiler settings for all stored objects in the database.
- USER_PLSQL_OBJECT_SETTINGS displays information about the compiler settings for the stored objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| TYPE | VARCHAR2(12) | | Type of the object:<br>• PROCEDURE<br>• FUNCTION<br>• PACKAGE<br>• PACKAGE BODY<br>• TRIGGER<br>• TYPE<br>• TYPE BODY |
| PLSQL_OPTIMIZE_LEVEL | NUMBER | | Optimization level that was used to compile the object |
| PLSQL_CODE_TYPE | VARCHAR2(4000) | | Compilation mode for the object |
| PLSQL_DEBUG | VARCHAR2(4000) | | Indicates whether the object was compiled with debug information or not |
| PLSQL_WARNINGS | VARCHAR2(4000) | | Compiler warning settings that were used to compile the object |
| NLS_LENGTH_SEMANTICS | VARCHAR2(4000) | | NLS length semantics that were used to compile the object |
| PLSQL_CCFLAGS | VARCHAR2(4000) | | Conditional compilation flag settings that were used to compile the object |
| PLSCOPE_SETTINGS | VARCHAR2(4000) | | Settings for using PL/Scope |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ORIGIN_CON_ID | NUMBER | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_PLSQL_OBJECT_SETTINGS"
- "USER_PLSQL_OBJECT_SETTINGS"

# ALL_PLSQL_TYPE_ATTRS

ALL_PLSQL_TYPE_ATTRS describes the attributes of PL/SQL types accessible to the user.

### Related Views

- DBA_PLSQL_TYPE_ATTRS describes the attributes of all PL/SQL types in the database.
- USER_PLSQL_TYPE_ATTRS describes the attributes of the user's own PL/SQL types. This view does not display the OWNER or CHAR_USED columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the type |
| TYPE_NAME | VARCHAR2(136) | | Name of the type |
| PACKAGE_NAME | VARCHAR2(128) | | Name of the package containing the type |
| ATTR_NAME | VARCHAR2(128) | | Name of the attribute |
| ATTR_TYPE_MOD | VARCHAR2(7) | | Type modifier of the attribute |
| ATTR_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the attribute |
| ATTR_TYPE_NAME | VARCHAR2(136) | | Name of the type of the attribute |
| ATTR_TYPE_PACKAGE | VARCHAR2(128) | | Name of the package containing the attribute type |
| LENGTH | NUMBER | | Length of the CHAR attribute or maximum length of the VARCHAR or VARCHAR2 attribute |
| PRECISION | NUMBER | | Decimal precision of the NUMBER or DECIMAL attribute or binary precision of the FLOAT attribute |
| SCALE | NUMBER | | Scale of the NUMBER or DECIMAL attribute |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CHARACTER_SET_NAME | VARCHAR2(44) | | Character set name of the attribute |
| ATTR_NO | NUMBER | | Syntactical order number or position of the attribute as specified in the type specification or CREATE TYPE statement (not to be used as ID number) |
| CHAR_USED | VARCHAR2(1) | | C if the width was specified in characters, B if in bytes |

**See Also:**

- "DBA_PLSQL_TYPE_ATTRS"
- "USER_PLSQL_TYPE_ATTRS"

# ALL_PLSQL_TYPES

ALL_PLSQL_TYPES describes the PL/SQL types accessible to the user.

### Related Views

- DBA_PLSQL_TYPES describes all the PL/SQL types in the database.
- USER_PLSQL_TYPES describes the user's own PL/SQL types. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the type |
| TYPE_NAME | VARCHAR2(136) | | Name of the type |
| PACKAGE_NAME | VARCHAR2(128) | NOT NULL | Name of the package containing the type |
| TYPE_OID | RAW(16) | NOT NULL | Object identifier (OID) of the type |
| TYPECODE | VARCHAR2(58) | | Typecode of the type |
| ATTRIBUTES | NUMBER | | Number of attributes in the type |
| CONTAINS_PLSQL | VARCHAR2(3) | | Indicates whether the type contains PL/SQL-specific data types (YES) or not (NO) |

**See Also:**

- "DBA_PLSQL_TYPES"
- "USER_PLSQL_TYPES"

# ALL_POLICIES

ALL_POLICIES describes all Oracle Virtual Private Database (VPD) security policies for objects accessible to the current user. A security policy is a list of security requirements and rules that regulate row level access to those database objects.

### Related Views

- DBA_POLICIES describes all Oracle Virtual Private Database (VPD) security policies in the database.
- USER_POLICIES describes all Oracle Virtual Private Database (VPD) security policies associated with objects owned by the current user. This view does not display the OBJECT_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the synonym, table, or view |
| OBJECT_NAME | VARCHAR2(128) | | Name of the synonym, table, or view |
| POLICY_GROUP | VARCHAR2(128) | | Name of the policy group |
| POLICY_NAME | VARCHAR2(128) | | Name of the policy |
| PF_OWNER | VARCHAR2(128) | | Owner of the policy function |
| PACKAGE | VARCHAR2(128) | | Name of the package containing the policy function |
| FUNCTION | VARCHAR2(128) | | Name of the policy function |
| SEL | VARCHAR2(3) | | Indicates whether the policy is applied to queries on the object (YES) or not (NO) |
| INS | VARCHAR2(3) | | Indicates whether the policy is applied to INSERT statements on the object (YES) or not (NO) |
| UPD | VARCHAR2(3) | | Indicates whether the policy is applied to UPDATE statements on the object (YES) or not (NO) |
| DEL | VARCHAR2(3) | | Indicates whether the policy is applied to DELETE statements on the object (YES) or not (NO) |
| IDX | VARCHAR2(3) | | Indicates whether the policy is enforced for index maintenance on the object (YES) or not (NO) |
| CHK_OPTION | VARCHAR2(3) | | Indicates whether the check option is enforced for the policy (YES) or not (NO) |
| ENABLE | VARCHAR2(3) | | Indicates whether the policy is enabled (YES) or disabled (NO) |
| STATIC_POLICY | VARCHAR2(3) | | Indicates whether the policy is static (YES) or not (NO). This column is obsolete because information |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | about static policies is shown in the POLICY_TYPE column. |
| POLICY_TYPE | VARCHAR2(24) | | Policy type:<br>• STATIC<br>• SHARED_STATIC<br>• CONTEXT_SENSITIVE<br>• SHARED_CONTEXT_SENSITIVE<br>• DYNAMIC |
| LONG_PREDICATE | VARCHAR2(3) | | Indicates whether the policy function can return a maximum of 32 KB of predicate (YES) or not (NO). If NO, the default maximum predicate size is 4000 bytes. |

**See Also:**

- "DBA_POLICIES"
- "USER_POLICIES"
- *Oracle Database Concepts* for an overview of security policies and fine-grained access control
- *Oracle Database Security Guide* for more information about security policies
- The DBMS_RLS package in *Oracle Database PL/SQL Packages and Types Reference* for information on administering security policies

# ALL_POLICY_ATTRIBUTES

ALL_POLICY_ATTRIBUTES lists the attribute associations {Namespaces, Attributes} of context-sensitive and shared context-sensitive Oracle Virtual Private Database (VPD) policies for objects accessible to the current user.

### Related Views

- DBA_POLICY_ATTRIBUTES lists the attribute associations {Namespaces, Attributes} of all context-sensitive and shared context-sensitive Oracle Virtual Private Database (VPD) policies in the database.
- USER_POLICY_ATTRIBUTES lists the attribute associations {Namespaces, Attributes} of all context-sensitive and shared-context sensitive Oracle Virtual Private Database (VPD) policies for synonyms, tables, or views owned by the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the synonym, table, or view |
| OBJECT_NAME | VARCHAR2(128) | | Name of the synonym, table, or view |
| POLICY_GROUP | VARCHAR2(128) | | Name of the policy group |
| POLICY_NAME | VARCHAR2(128) | | Name of the policy |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAMESPACE | VARCHAR2(128) | | Name of the local application context |
| ATTRIBUTE | VARCHAR2(128) | | Name of the attribute |

**See Also:**

- "DBA_POLICY_ATTRIBUTES"
- "USER_POLICY_ATTRIBUTES"

# ALL_POLICY_CONTEXTS

ALL_POLICY_CONTEXTS describes the driving contexts defined for the synonyms, tables, and views accessible to the current user.

### Related Views

- DBA_POLICY_CONTEXTS describes all driving contexts in the database.
- USER_POLICY_CONTEXTS describes the driving contexts defined for the synonyms, tables, and views owned by the current user. This view does not display the OBJECT_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the synonym, table, or view |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the synonym, table, or view |
| NAMESPACE | VARCHAR2(128) | NOT NULL | Namespace of the driving context |
| ATTRIBUTE | VARCHAR2(128) | NOT NULL | Attribute of the driving context |

**See Also:**

- "DBA_POLICY_CONTEXTS"
- "USER_POLICY_CONTEXTS"

# ALL_POLICY_GROUPS

ALL_POLICY_GROUPS describes the policy groups defined for the synonyms, tables, and views accessible to the current user.

### Related Views

- DBA_POLICY_GROUPS describes all policy groups in the database.
- USER_POLICY_GROUPS describes the policy groups defined for the synonyms, tables, and views owned by the current user. This view does not display the OBJECT_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the synonym, table, or view |
| OBJECT_NAME | VARCHAR2(128) | | Name of the synonym, table, or view |
| POLICY_GROUP | VARCHAR2(128) | | Name of the policy group |

**See Also:**

- "DBA_POLICY_GROUPS"
- "USER_POLICY_GROUPS"

# ALL_PROCEDURES

ALL_PROCEDURES lists all functions and procedures that are accessible to the current user, along with associated properties. For example, ALL_PROCEDURES indicates whether or not a function is pipelined, parallel enabled or an aggregate function. If a function is pipelined or an aggregate function, the associated implementation type (if any) is also identified.

### Related Views

- DBA_PROCEDURES lists all functions and procedures available in the database, along with associated properties.
- USER_PROCEDURES lists all functions and procedures owned by the current user, along with associated properties. It does not contain the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the procedure |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object: top-level function, procedure, or package name |
| PROCEDURE_NAME | VARCHAR2(128) | | Name of the procedure |
| OBJECT_ID | NUMBER | | Object number of the object |
| SUBPROGRAM_ID | NUMBER | | Unique subprogram identifier |
| OVERLOAD | VARCHAR2(40) | | Overload unique identifier |
| OBJECT_TYPE | VARCHAR2(13) | | The typename of the object |
| AGGREGATE | VARCHAR2(3) | | Indicates whether the procedure is an aggregate function (YES) or not (NO) |
| PIPELINED | VARCHAR2(3) | | Indicates whether the procedure is a pipelined table function (YES) or not (NO) |
| IMPLTYPEOWNER | VARCHAR2(128) | | Owner of the implementation type, if any |
| IMPLTYPENAME | VARCHAR2(128) | | Name of the implementation type, if any |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARALLEL | VARCHAR2(3) | | Indicates whether the procedure or function is parallel-enabled (YES) or not (NO) |
| INTERFACE | VARCHAR2(3) | | YES, if the procedure/function is a table function implemented using the ODCI interface; otherwise NO |
| DETERMINISTIC | VARCHAR2(3) | | YES, if the procedure/function is declared to be deterministic; otherwise NO |
| AUTHID | VARCHAR2(12) | | Indicates whether the procedure/function is declared to execute as DEFINER or CURRENT_USER (invoker) |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_PROCEDURES"
- "USER_PROCEDURES"
- "ALL_ARGUMENTS" for information about the arguments of the functions and procedures that are accessible to the current user

# ALL_PROPAGATION

ALL_PROPAGATION displays information about the Streams propagations that have a source queue accessible to the current user.

### Related View

DBA_PROPAGATION displays information about all Streams propagations in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROPAGATION_NAME | VARCHAR2(128) | NOT NULL | Name of the Streams propagation |
| SOURCE_QUEUE_OWNER | VARCHAR2(128) | | Owner of the source queue of the propagation |
| SOURCE_QUEUE_NAME | VARCHAR2(128) | | Name of the source queue of the propagation |
| DESTINATION_QUEUE_OWNER | VARCHAR2(128) | | Owner of the destination queue of the propagation |
| DESTINATION_QUEUE_NAME | VARCHAR2(128) | | Name of the destination queue of the propagation |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DESTINATION_DBLINK | VARCHAR2(128) | | Database link to propagate events from the source queue to the destination queue |
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the propagation positive rule set |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the propagation positive rule set |
| NEGATIVE_RULE_SET_OWNER | VARCHAR2(128) | | Owner of the propagation negative rule set |
| NEGATIVE_RULE_SET_NAME | VARCHAR2(128) | | Name of the propagation negative rule set |
| QUEUE_TO_QUEUE | VARCHAR2(5) | | Indicates whether the propagation is a queue-to-queue propagation (TRUE) or not (FALSE). A queue-to-queue propagation always has its own exclusive propagation job to propagate messages from the source queue to the destination queue. |
| STATUS | VARCHAR2(8) | | Status of the propagation:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message last encountered by propagation |
| ERROR_DATE | DATE | | Time that propagation last encountered an error |
| ORIGINAL_PROPAGATION_NAME | VARCHAR2(128) | | Original propagation from which the propagation is cloned |
| ORIGINAL_SOURCE_QUEUE_OWNER | VARCHAR2(128) | | Source queue owner of the original propagation |
| ORIGINAL_SOURCE_QUEUE_NAME | VARCHAR2(128) | | Source queue name of the original propagation |
| ACKED_SCN | NUMBER | | Acknowledged SCN of the subscribers of captured messages in the destination queue for the propagation |
| AUTO_MERGE_THRESHOLD | NUMBER | | Merge threshold value for merging the propagation back to the original source queue. This is used as the value for the merge_threshold parameter in the DBMS_STREAMS_ADM.MERGE_STREAMS_JOB procedure. |

**See Also:**

- "DBA_PROPAGATION"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STREAMS_ADM.MERGE_STREAMS_JOB procedure

# ALL_QUEUE_SCHEDULES

ALL_QUEUE_SCHEDULES describes the propagation schedules whose source queues are accessible to the current user.

### Related Views

- DBA_QUEUE_SCHEDULES describes all propagation schedules in the database.
- USER_QUEUE_SCHEDULES describes the propagation schedules whose source queues are owned by the current user. This view does not display the SCHEMA column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA | VARCHAR2(128) | | Source queue owner |
| QNAME | VARCHAR2(128) | | Source queue name |
| DESTINATION | VARCHAR2(128) | | Destination name, currently limited to be a DBLINK name |
| START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date at which to start propagation |
| START_TIME | VARCHAR2(8) | | Time of day at which to start propagation (in HH:MI:SS format) |
| PROPAGATION_WINDOW | NUMBER | | Duration for the propagation window (in seconds) |
| NEXT_TIME | VARCHAR2(4000) | | Function to compute the start of the next propagation window |
| LATENCY | NUMBER | | Maximum wait time to propagate a message during the propagation window |
| SCHEDULE_DISABLED | VARCHAR2(1) | | Indicates whether the schedule is disabled (Y) or enabled (N). If disabled, then the schedule will not be executed. |
| PROCESS_NAME | VARCHAR2(4) | | Name of the process executing the schedule; NULL if not currently executing |
| SESSION_ID | VARCHAR2(82) | | Session ID and session serial number of the job executing this schedule (SID, SERIAL#); NULL if not currently executing |
| INSTANCE | NUMBER | | Cluster database instance number executing the schedule |
| LAST_RUN_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date of the last successful execution |
| LAST_RUN_TIME | VARCHAR2(8) | | Time of day of the last successful execution (in HH:MI:SS format) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CURRENT_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date at which the current window of this schedule was started |
| CURRENT_START_TIME | VARCHAR2(8) | | Time of day at which the current window of this schedule was started (in HH:MI:SS format) |
| NEXT_RUN_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date at which the next window of this schedule will be started |
| NEXT_RUN_TIME | VARCHAR2(8) | | Time of day at which the next window of this schedule will be started (in HH:MI:SS format) |
| TOTAL_TIME | NUMBER | | Total time spent by the system in executing this schedule (in seconds) |
| TOTAL_NUMBER | NUMBER | | Total number of messages propagated in this schedule |
| TOTAL_BYTES | NUMBER | | Total number of bytes propagated in this schedule |
| MAX_NUMBER | NUMBER | | Maximum number of messages propagated in a propagation window |
| MAX_BYTES | NUMBER | | Maximum number of bytes propagated in a propagation window |
| AVG_NUMBER | NUMBER | | Average number of messages propagated in a propagation window |
| AVG_SIZE | NUMBER | | Average size of a propagated message (in bytes) |
| AVG_TIME | NUMBER | | Average time to propagate a message (in seconds) |
| FAILURES | NUMBER | | Number of consecutive times schedule execution has failed, if any. After 16 consecutive failures, a propagation job becomes disabled automatically. |
| LAST_ERROR_DATE | DATE | | Date of the last unsuccessful execution |
| LAST_ERROR_TIME | VARCHAR2(8) | | Time of day of the last unsuccessful execution (in HH:MI:SS format) |
| LAST_ERROR_MSG | VARCHAR2(4000) | | Error number and error message text of the last unsuccessful execution |
| MESSAGE_DELIVERY_MODE | VARCHAR2(10) | | Message delivery mode:<br>• PERSISTENT<br>• BUFFERED |
| ELAPSED_DEQUEUE_TIME | NUMBER | | Elapsed dequeue time (in hundredths of a second) |
| ELAPSED_PICKLE_TIME | NUMBER | | Elapsed pickle time (time taken to linearize a logical change record (LCR) into a stream of bytes that can be sent over the network) (in hundredths of a second) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_NAME | VARCHAR2(128) | | Name of the Scheduler job |

**See Also:**

- "DBA_QUEUE_SCHEDULES"
- "USER_QUEUE_SCHEDULES"

# ALL_QUEUE_SUBSCRIBERS

ALL_QUEUE_SUBSCRIBERS displays the list of subscribers that the current user has privilege to dequeue from.

### Related Views

- DBA_QUEUE_SUBSCRIBERS displays the list of subscribers on all queues in the database.
- USER_QUEUE_SUBSCRIBERS displays the list of subscribers on queues that are under the current user's schema. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| QUEUE_TABLE | VARCHAR2(128) | NOT NULL | Name of the queue table on which the queue is defined |
| CONSUMER_NAME | VARCHAR2(30) | | Name of the subscriber |
| ADDRESS | VARCHAR2(1024) | | Address of the subscriber |
| PROTOCOL | NUMBER | | Protocol of the subscriber |
| TRANSFORMATION | VARCHAR2(61) | | Transformation for the subscriber |
| RULE | CLOB | | Rule condition for the subscriber |
| DELIVERY_MODE | VARCHAR2(22) | | Message delivery mode for the subscriber:<br>• PERSISTENT<br>• BUFFERED<br>• PERSISTENT_OR_BUFFERED |
| IF_NONDURABLE_SUBSCRIBER | VARCHAR2(3) | | Indicates whether the subscriber is a non-durable subscriber (YES) or not (NO) |
| QUEUE_TO_QUEUE | VARCHAR2(5) | | Indicates whether the subscriber is a queue-to-queue subscriber (TRUE) or not (FALSE) |
| SUBSCRIBER_ID | NUMBER | | ID of the subscriber |
| POS_BITMAP | NUMBER | | Position of the subscriber in the bitmap |

**See Also:**

- "DBA_QUEUE_SUBSCRIBERS"
- "USER_QUEUE_SUBSCRIBERS"

# ALL_QUEUE_TABLES

ALL_QUEUE_TABLES describes the queues in the queue tables accessible to the current user.

### Related Views

- DBA_QUEUE_TABLES describes the queues in all queue tables in the database.
- USER_QUEUE_TABLES describes the queues in the queue tables created in the current user's schema. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the queue table |
| QUEUE_TABLE | VARCHAR2(128) | | Name of the queue table |
| TYPE | VARCHAR2(9) | | Type of user data:<br>• RAW - Raw type<br>• OBJECT - User-defined object type<br>• VARIANT - Variant type (for internal use only) |
| OBJECT_TYPE | VARCHAR2(257) | | Object type of the payload when TYPE is OBJECT |
| SORT_ORDER | VARCHAR2(22) | | User-specified sort order |
| RECIPIENTS | VARCHAR2(8) | | SINGLE or MULTIPLE recipients |
| MESSAGE_GROUPING | VARCHAR2(13) | | NONE or TRANSACTIONAL |
| COMPATIBLE | VARCHAR2(6) | | Lowest release level which the queue table is compatible with (for example, 8.0.3) |
| PRIMARY_INSTANCE | NUMBER | | Indicates the instance number of the instance which is the primary owner of the queue table. A value of 0 indicates that there is no primary owner. |
| SECONDARY_INSTANCE | NUMBER | | Indicates the instance number of the instance which is the secondary owner of the queue table. This instance becomes the owner of the queue table if the primary owner is not alive. A value of 0 indicates that there is no secondary owner. |
| OWNER_INSTANCE | NUMBER | | Instance number of the instance which currently owns the queue table |
| USER_COMMENT | VARCHAR2(50) | | Comment supplied by the user |
| SECURE | VARCHAR2(3) | | Indicates whether the queue table is secure (YES) or not (NO) |

**See Also:**

- "DBA_QUEUE_TABLES"
- "USER_QUEUE_TABLES"
- *Oracle Database Advanced Queuing User's Guide* for more information Advanced Queuing

# ALL_QUEUES

ALL_QUEUES describes all queues on which the current user has enqueue or dequeue privileges. If the user has any Advanced Queuing system privileges, like MANAGE ANY QUEUE, ENQUEUE ANY QUEUE or DEQUEUE ANY QUEUE, then this view describes all queues in the database.

### Related Views

- DBA_QUEUES describes all queues in the database.
- USER_QUEUES describes the operational characteristics of every queue owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue |
| NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| QUEUE_TABLE | VARCHAR2(128) | NOT NULL | Name of the table the queue data resides in |
| QID | NUMBER | NOT NULL | Object number of the queue |
| QUEUE_TYPE | VARCHAR2(20) | | Type of the queue:<br>• EXCEPTION_QUEUE<br>• NON_PERSISTENT_QUEUE<br>• NORMAL_QUEUE |
| MAX_RETRIES | NUMBER | | Maximum number of retries allowed when dequeuing from the queue |
| RETRY_DELAY | NUMBER | | Time interval between retries |
| ENQUEUE_ENABLED | VARCHAR2(7) | | Indicates whether the queue is enabled for enqueue (YES) or not (NO) |
| DEQUEUE_ENABLED | VARCHAR2(7) | | Indicates whether the queue is enabled for dequeue (YES) or not (NO) |
| RETENTION | VARCHAR2(40) | | Time interval (in seconds) processed messages are retained in the queue, or FOREVER |
| USER_COMMENT | VARCHAR2(50) | | User specified comment |
| NETWORK_NAME | VARCHAR2(512) | | Network name |
| SHARDED | VARCHAR2(5) | | TRUE if queue is sharded, FALSE otherwise |

**See Also:**

- "DBA_QUEUES"
- "USER_QUEUES"
- *Oracle Database Advanced Queuing User's Guide* for more information Advanced Queuing

# ALL_REFRESH

ALL_REFRESH describes all the refresh groups accessible to the current user.

### Related Views

- DBA_REFRESH describes all refresh groups in the database.
- USER_REFRESH describes all refresh groups owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROWNER | VARCHAR2(128) | NOT NULL | Owner of the refresh group |
| RNAME | VARCHAR2(128) | NOT NULL | Name of the refresh group |
| REFGROUP | NUMBER | | Internal identifier of the refresh group |
| IMPLICIT_DESTROY | VARCHAR2(1) | | Indicates whether the refresh group is destroyed when its last item is subtracted (Y) or not (N) |
| PUSH_DEFERRED_RPC | VARCHAR2(1) | | Indicates whether changes are pushed from the snapshot to the master before refresh (Y) or not (N) |
| REFRESH_AFTER_ERRORS | VARCHAR2(1) | | Indicates whether to proceed with refresh despite errors when pushing deferred RPCs (Y) or not (N) |
| ROLLBACK_SEG | VARCHAR2(128) | | Name of the rollback segment to use while refreshing |
| JOB | NUMBER | | Identifier of the job used to refresh the group automatically |
| NEXT_DATE | DATE | | Date that this job will next be refreshed automatically, if not broken |
| INTERVAL | VARCHAR2(200) | | A date function used to compute the next NEXT_DATE |
| BROKEN | VARCHAR2(1) | | Indicates whether the job is broken and will never be run (Y) or not (N) |
| PURGE_OPTION | NUMBER(38) | | Method for purging the transaction queue after each push (1 indicates quick purge option; 2 indicates precise purge option) |
| PARALLELISM | NUMBER(38) | | Level of parallelism for transaction propagation |
| HEAP_SIZE | NUMBER(38) | | Size of the heap |

**See Also:**

- "DBA_REFRESH"
- "USER_REFRESH"

# ALL_REFRESH_CHILDREN

ALL_REFRESH_CHILDREN describes all the objects in refresh groups that are accessible to the current user.

### Related Views

- DBA_REFRESH_CHILDREN describes the objects in all refresh groups in the database.
- USER_REFRESH_CHILDREN describes the objects in all refresh groups owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object in the refresh group |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object in the refresh group |
| TYPE | VARCHAR2(128) | | Type of the object in the refresh group |
| ROWNER | VARCHAR2(128) | NOT NULL | Owner of the refresh group |
| RNAME | VARCHAR2(128) | NOT NULL | Name of the refresh group |
| REFGROUP | NUMBER | | Internal identifier of the refresh group |
| IMPLICIT_DESTROY | VARCHAR2(1) | | Indicates whether the refresh group is destroyed when its last item is subtracted (Y) or not (N) |
| PUSH_DEFERRED_RPC | VARCHAR2(1) | | Indicates whether changes are pushed from the snapshot to the master before refresh (Y) or not (N) |
| REFRESH_AFTER_ERRORS | VARCHAR2(1) | | Indicates whether to proceed with refresh despite errors when pushing deferred RPCs (Y) or not (N) |
| ROLLBACK_SEG | VARCHAR2(128) | | Name of the rollback segment to use while refreshing |
| JOB | NUMBER | | Identifier of the job used to refresh the group automatically |
| NEXT_DATE | DATE | | Date that this job will next be refreshed automatically, if not broken |
| INTERVAL | VARCHAR2(200) | | A date function used to compute the next NEXT_DATE |
| BROKEN | VARCHAR2(1) | | Indicates whether the job is broken and will never be run (Y) or not (N) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PURGE_OPTION | NUMBER(38) | | Method for purging the transaction queue after each push. 1 indicates quick purge option; 2 indicates precise purge option |
| PARALLELISM | NUMBER(38) | | Level of parallelism for transaction propagation |
| HEAP_SIZE | NUMBER(38) | | Size of the heap |

**See Also:**

- "DBA_REFRESH_CHILDREN"
- "USER_REFRESH_CHILDREN"

# ALL_REFRESH_DEPENDENCIES

ALL_REFRESH_DEPENDENCIES displays the names of the dependent detail or container tables of all the materialized views in the current schema.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Table name, unique within this schema |
| PARENT_OBJECT_TYPE | CHAR(17) | | MATERIALIZED VIEW |
| OLDEST_REFRESH_SCN | NUMBER | | Minimum SCN of any summary or materialized view that has TABLE_NAME as a detail table |
| OLDEST_REFRESH_DATE | DATE | | SYSDATE when last refreshed |

# ALL_REFS

ALL_REFS describes the REF columns and REF attributes in object type columns accessible to the current user.

### Related Views

- DBA_REFS describes all REF columns and REF attributes in the database.
- USER_REFS describes the REF columns and REF attributes in object type columns owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the REF column or attribute. If it is not a top-level attribute, the value of COLUMN_NAME |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | should be a path name starting with the column name. |
| WITH_ROWID | VARCHAR2(3) | | Indicates whether the REF value is stored with ROWID (YES) or not (NO) |
| IS_SCOPED | VARCHAR2(3) | | Indicates whether the REF column is scoped (YES) or not (NO) |
| SCOPE_TABLE_OWNER | VARCHAR2(128) | | Owner of the scope table, if it exists and is accessible by the user |
| SCOPE_TABLE_NAME | VARCHAR2(128) | | Name of the scope table, if it exists and is accessible by the user |
| OBJECT_ID_TYPE | VARCHAR2(33) | | Indicates whether the object ID (OID) is USER-DEFINED or SYSTEM GENERATED |

**See Also:**

- "DBA_REFS"
- "USER_REFS"

# ALL_REGISTERED_MVIEWS

ALL_REGISTERED_MVIEWS describes all registered materialized views (registered at a master site or a master materialized view site) accessible to the current user.

A materialized view created with the BUILD DEFERRED option of the CREATE MATERIALIZED VIEW statement is only registered with ALL_REGISTERED_MVIEWS if that materialized view has been completely refreshed at least once.

### Related Views

- DBA_REGISTERED_MVIEWS describes all registered materialized views in the database.
- USER_REGISTERED_MVIEWS describes all registered materialized views owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the materialized view |
| NAME | VARCHAR2(128) | NOT NULL | Name of the materialized view |
| MVIEW_SITE | VARCHAR2(128) | NOT NULL | Global name of the materialized view site |
| CAN_USE_LOG | VARCHAR2(3) | | Indicates whether the materialized view can use a materialized view log (YES) or the materialized view is too complex to use a log (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| UPDATABLE | VARCHAR2(3) | | Indicates whether the materialized view is updatable (YES) or not and the materialized view is read only (NO) |
| REFRESH_METHOD | VARCHAR2(11) | | Indicates whether the materialized view uses primary key (PRIMARY KEY), rowids (ROWID), or object identifiers (OBJECT ID) for fast refresh |
| MVIEW_ID | NUMBER(38) | | Identifier for the materialized view used by the masters for fast refresh |
| VERSION | VARCHAR2(26) | | Oracle version of the materialized view<br>**Note:** Oracle Database materialized views show ORACLE 8 MATERIALIZED VIEW. |
| QUERY_TXT | LONG | | Query that defines the materialized view |

**See Also:**

- "DBA_REGISTERED_MVIEWS"
- "USER_REGISTERED_MVIEWS"

# ALL_REGISTRY_BANNERS

ALL_REGISTRY_BANNERS displays the valid components loaded into the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BANNER | VARCHAR2(80) | | Component display banner |

# ALL_REPL_DBNAME_MAPPING

ALL_REPL_DBNAME_MAPPING provides details about the database name mapping in replication for the current user.

### Related View

DBA_REPL_DBNAME_MAPPING provides details about the database name mapping in replication.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The fully qualified global name of the root in a multitenant container database (CDB) where the changes originated |
| SOURCE_DATABASE_NAME | VARCHAR2(128) | | The fully qualified global name of the pluggable database (PDB) where the changes originated |
| SOURCE_CONTAINER_NAME | VARCHAR2(128) | | The container name of the database where the changes originated |

**See Also:**

"DBA_REPL_DBNAME_MAPPING"

# ALL_REWRITE_EQUIVALENCES

ALL_REWRITE_EQUIVALENCES describes the rewrite equivalences accessible to the current user.

### Related Views

- DBA_REWRITE_EQUIVALENCES describes all rewrite equivalences in the database.
- USER_REWRITE_EQUIVALENCES describes the rewrite equivalences owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the rewrite equivalence |
| NAME | VARCHAR2(128) | NOT NULL | Name of the rewrite equivalence |
| SOURCE_STMT | CLOB | | Source statement of the rewrite equivalence |
| DESTINATION_STMT | CLOB | | Destination of the rewrite equivalence |
| REWRITE_MODE | VARCHAR2(10) | | Rewrite mode of the rewrite equivalence:<br>• DISABLED<br>• TEXT_MATCH<br>• GENERAL<br>• RECURSIVE |

**See Also:**

- "DBA_REWRITE_EQUIVALENCES"
- "USER_REWRITE_EQUIVALENCES"

# ALL_RULE_SET_RULES

ALL_RULE_SET_RULES describes the rules in the rule sets accessible to the current user.

### Related Views

- DBA_RULE_SET_RULES describes the rules in all rule sets in the database.
- USER_RULE_SET_RULES describes the rules in the rule sets owned by the current user. This view does not display the RULE_SET_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_SET_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule set |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_SET_NAME | VARCHAR2(128) | NOT NULL | Name of the rule set |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_SET_RULE_ENABLED | VARCHAR2(8) | | Indicates whether the rule is enabled in the rule set (ENABLED) or not (DISABLED) |
| RULE_SET_RULE_EVAL_CTX_OWNER | VARCHAR2(128) | | Owner of the evaluation context specified when the rule was added to the rule set, if any |
| RULE_SET_RULE_EVAL_CTX_NAME | VARCHAR2(128) | | Name of the evaluation context specified when the rule was added to the rule set, if any |
| RULE_SET_RULE_COMMENT | VARCHAR2(4000) | | Comment specified when the rule was added to the rule set, if any |

**See Also:**

- "DBA_RULE_SET_RULES"
- "USER_RULE_SET_RULES"

# ALL_RULE_SETS

ALL_RULE_SETS describes the rule sets accessible to the current user.

### Related Views

- DBA_RULE_SETS describes all rule sets in the database.
- USER_RULE_SETS describes the rule sets owned by the current user. This view does not display the RULE_SET_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_SET_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule set |
| RULE_SET_NAME | VARCHAR2(128) | NOT NULL | Name of the rule set |
| RULE_SET_EVAL_CONTEXT_OWNER | VARCHAR2(128) | | Owner of the evaluation context associated with the rule set, if any |
| RULE_SET_EVAL_CONTEXT_NAME | VARCHAR2(128) | | Name of the evaluation context associated with the rule set, if any |
| RULE_SET_COMMENT | VARCHAR2(4000) | | Comment specified with the rule set, if any |

**See Also:**

- "DBA_RULE_SETS"
- "USER_RULE_SETS"

# ALL_RULES

ALL_RULES describes the rules accessible to the current user.

### Related Views

- DBA_RULES describes all rules in the database.
- USER_RULES describes the rules owned by the current user. This view does not display the RULE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_CONDITION | CLOB | | Expressions and operators that constitute the rule condition |
| RULE_EVALUATION_CONTEXT_OWNER | VARCHAR2(128) | | Owner of the evaluation context associated with the rule, if any |
| RULE_EVALUATION_CONTEXT_NAME | VARCHAR2(128) | | Name of the evaluation context associated with the rule, if any |
| RULE_ACTION_CONTEXT | RE$NV_LIST | | Action context associated with the rule, if any |
| RULE_COMMENT | VARCHAR2(4000) | | Comment specified with the rule, if any |

**See Also:**

- "DBA_RULES"
- "USER_RULES"

# ALL_SCHEDULER_CHAIN_RULES

ALL_SCHEDULER_CHAIN_RULES displays information about the rules for the chains accessible to the current user (that is, those chains that the user has ALTER or EXECUTE privileges for).

### Related Views

- DBA_SCHEDULER_CHAIN_RULES displays information about the rules for all chains in the database.

- USER_SCHEDULER_CHAIN_RULES displays information about the rules for the chains owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler chain that the rule is in |
| CHAIN_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler chain that the rule is in |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | | Name of the rule |
| CONDITION | VARCHAR2(4000) | | Boolean condition triggering the rule |
| ACTION | VARCHAR2(4000) | | Action to be performed when the rule is triggered |
| COMMENTS | VARCHAR2(4000) | | User-specified comments about the rule |

**See Also:**

- "DBA_SCHEDULER_CHAIN_RULES"
- "USER_SCHEDULER_CHAIN_RULES"

# ALL_SCHEDULER_CHAIN_STEPS

ALL_SCHEDULER_CHAIN_STEPS displays information about the defined steps of the chains accessible to the current user (that is, those chains that the user has ALTER or EXECUTE privileges for).

### Related Views

- DBA_SCHEDULER_CHAIN_STEPS displays information about the defined steps of all chains in the database.
- USER_SCHEDULER_CHAIN_STEPS displays information about the defined steps of the chains owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler chain the step is in |
| CHAIN_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler chain the step is in |
| STEP_NAME | VARCHAR2(128) | NOT NULL | Name of the chain step |
| PROGRAM_OWNER | VARCHAR2(392) | | Owner of the program that runs during the step |
| PROGRAM_NAME | VARCHAR2(392) | | Name of the program that runs during the step |
| EVENT_SCHEDULE_OWNER | VARCHAR2(392) | | Owner of the event schedule that this step waits for |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVENT_SCHEDULE_NAME | VARCHAR2(392) | | Name of the event schedule that this step waits for |
| EVENT_QUEUE_OWNER | VARCHAR2(128) | | Owner of the source queue into which the event will be raised |
| EVENT_QUEUE_NAME | VARCHAR2(128) | | Name of the source queue into which the event will be raised |
| EVENT_QUEUE_AGENT | VARCHAR2(128) | | Name of the AQ agent used by the user on the event source queue (for a secure queue) |
| EVENT_CONDITION | VARCHAR2(4000) | | Boolean expression used as the subscription rule for an event on the source queue |
| CREDENTIAL_OWNER | VARCHAR2(128) | | Owner of the credential to be used for an external step job |
| CREDENTIAL_NAME | VARCHAR2(128) | | Name of the credential to be used for an external step job |
| DESTINATION | VARCHAR2(128) | | Destination host on which a remote step job will run |
| SKIP | VARCHAR2(5) | | Indicates whether the step should be skipped (TRUE) or not (FALSE) |
| PAUSE | VARCHAR2(5) | | Indicates whether the step should be paused after running (TRUE) or not (FALSE) |
| PAUSE_BEFORE | VARCHAR2(5) | | Indicates whether the step should be paused before running (TRUE) or not (FALSE) |
| RESTART_ON_RECOVERY | VARCHAR2(5) | | Indicates whether the step should be restarted on database recovery (TRUE) or not (FALSE) |
| RESTART_ON_FAILURE | VARCHAR2(5) | | Indicates whether the step should be restarted on application failure (TRUE) or not (FALSE) |
| STEP_TYPE | VARCHAR2(21) | | Type of the step:<br>• EVENT_SCHEDULE<br>• INLINE_EVENT<br>• SUBCHAIN<br>• PROGRAM |
| TIMEOUT | INTERVAL DAY(3) TO SECOND(0) | | Timeout for waiting on an event schedule |

**See Also:**

- "DBA_SCHEDULER_CHAIN_STEPS"
- "USER_SCHEDULER_CHAIN_STEPS"

# ALL_SCHEDULER_CHAINS

ALL_SCHEDULER_CHAINS displays information about the chains accessible to the current user (that is, those chains that the user has ALTER or EXECUTE privileges for).

### Related Views

- DBA_SCHEDULER_CHAINS displays information about all chains in the database.
- USER_SCHEDULER_CHAINS displays information about the chains owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler chain |
| CHAIN_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler chain |
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the rule set describing the dependencies |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the rule set describing the dependencies |
| NUMBER_OF_RULES | NUMBER | | Number of rules in the chain |
| NUMBER_OF_STEPS | NUMBER | | Number of defined steps in the chain |
| ENABLED | VARCHAR2(5) | | Indicates whether the chain is enabled (TRUE) or disabled (FALSE) |
| EVALUATION_INTERVAL | INTERVAL DAY(3) TO SECOND(0) | | Periodic interval at which to reevaluate rules for the chain |
| USER_RULE_SET | VARCHAR2(5) | | Indicates whether the chain uses a user-specified rule set (TRUE) or not (FALSE) |
| COMMENTS | VARCHAR2(240) | | Comments on the chain |

**See Also:**

- "DBA_SCHEDULER_CHAINS"
- "USER_SCHEDULER_CHAINS"

# ALL_SCHEDULER_CREDENTIALS

ALL_SCHEDULER_CREDENTIALS displays information about the credentials accessible to the current user (that is, those credentials that the user has ALTER or EXECUTE privileges for).

**Note:**

This view is deprecated in favor of the ALL_CREDENTIALS view. Oracle recommends that you use ALL_CREDENTIALS instead. ALL_SCHEDULER_CREDENTIALS is retained for backward compatibility only.

### Related Views

- DBA_SCHEDULER_CREDENTIALS displays information about all credentials in the database.
- USER_SCHEDULER_CREDENTIALS displays information about the credentials owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler credential |
| CREDENTIAL_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler credential |
| USERNAME | VARCHAR2(128) | | Name of the user that will be used to log in to the remote database or operating system |
| DATABASE_ROLE | VARCHAR2(9) | | For a database target, the database role to use when logging in:<br>• SYSDBA<br>• SYSOPER |
| WINDOWS_DOMAIN | VARCHAR2(30) | | For a Windows target, the Windows domain to use when logging in |
| COMMENTS | VARCHAR2(240) | | Comments on the credential |

**See Also:**

- "ALL_CREDENTIALS"
- "DBA_SCHEDULER_CREDENTIALS"
- "USER_SCHEDULER_CREDENTIALS"

# ALL_SCHEDULER_DB_DESTS

ALL_SCHEDULER_DB_DESTS displays information about the destination objects accessible to the current user pointing to remote databases.

### Related Views

- DBA_SCHEDULER_DB_DESTS displays information about all destination objects in the database pointing to remote databases.
- USER_SCHEDULER_DB_DESTS displays information about the destination objects owned by the current user pointing to remote databases. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of this destination object |
| DESTINATION_NAME | VARCHAR2(128) | NOT NULL | Name of this destination object |
| CONNECT_INFO | VARCHAR2(4000) | | Connect string to connect to the remote database |
| AGENT | VARCHAR2(128) | | Name of the agent through which the connection to the remote database is being made |
| ENABLED | VARCHAR2(5) | | Indicates whether this destination object is enabled (TRUE) or disabled (FALSE) |
| REFS_ENABLED | VARCHAR2(5) | | Indicates whether all referenced objects are enabled (TRUE) or disabled (FALSE) |
| COMMENTS | VARCHAR2(240) | | Optional comment |

**See Also:**

- "DBA_SCHEDULER_DB_DESTS"
- "USER_SCHEDULER_DB_DESTS"

# ALL_SCHEDULER_DESTS

ALL_SCHEDULER_DESTS displays information about the destination objects for jobs accessible to the current user.

### Related Views

- DBA_SCHEDULER_DESTS displays information about all destination objects for jobs in the database.
- USER_SCHEDULER_DESTS displays information about the destination objects for jobs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of this destination object |
| DESTINATION_NAME | VARCHAR2(128) | NOT NULL | Name of this destination object |
| DESTINATION_TYPE | VARCHAR2(8) | | Type of this destination object:<br>• EXTERNAL<br>• DATABASE |
| ENABLED | VARCHAR2(5) | | Indicates whether this destination object is enabled (TRUE) or disabled (FALSE) |
| COMMENTS | VARCHAR2(240) | | Optional comment |

**See Also:**

- "DBA_SCHEDULER_DESTS"
- "USER_SCHEDULER_DESTS"

# ALL_SCHEDULER_EXTERNAL_DESTS

ALL_SCHEDULER_EXTERNAL_DESTS displays information about the destination objects accessible to the current user pointing to remote agents.

### Related View

DBA_SCHEDULER_EXTERNAL_DESTS displays information about all destination objects in the database pointing to remote agents.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of this destination object |
| DESTINATION_NAME | VARCHAR2(128) | NOT NULL | Name of this destination object |
| HOSTNAME | VARCHAR2(256) | | Name or IP address of the host on which the agent is located |
| PORT | NUMBER | | Port that the agent is listening on |
| IP_ADDRESS | VARCHAR2(64) | | IP address of the host on which the agent is located |
| ENABLED | VARCHAR2(5) | | Indicates whether this destination object is enabled (TRUE) or disabled (FALSE) |
| COMMENTS | VARCHAR2(240) | | Optional comment |

**See Also:**

"DBA_SCHEDULER_EXTERNAL_DESTS"

# ALL_SCHEDULER_FILE_WATCHERS

ALL_SCHEDULER_FILE_WATCHERS displays information about the Scheduler file watch requests accessible to the current user.

### Related Views

- DBA_SCHEDULER_FILE_WATCHERS displays information about all Scheduler file watch requests in the database.
- USER_SCHEDULER_FILE_WATCHERS displays information about the Scheduler file watch requests owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the file watch request |
| FILE_WATCHER_NAME | VARCHAR2(128) | NOT NULL | Name of the file watch request |
| ENABLED | VARCHAR2(5) | | Indicates whether this file watch request is enabled (TRUE) or disabled (FALSE) |
| DESTINATION_OWNER | VARCHAR2(128) | | Owner of the named destination object |
| DESTINATION | VARCHAR2(128) | | Name of the destination object |
| DIRECTORY_PATH | VARCHAR2(4000) | NOT NULL | Name of the directory path where the file will arrive |
| FILE_NAME | VARCHAR2(512) | NOT NULL | Name or pattern specifying the files that need to be monitored |
| CREDENTIAL_OWNER | VARCHAR2(128) | | Owner of the credential that should be used to authorize the file watch |
| CREDENTIAL_NAME | VARCHAR2(128) | | Name of the credential that should be used to authorize the file watch |
| MIN_FILE_SIZE | NUMBER | NOT NULL | Minimum size of the file being monitored |
| STEADY_STATE_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Time to wait before concluding that the file has stopped growing |
| LAST_MODIFIED_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time at which this file watcher was last modified |
| COMMENTS | VARCHAR2(240) | | Comments on the file watch request |

**See Also:**

- "DBA_SCHEDULER_FILE_WATCHERS"
- "USER_SCHEDULER_FILE_WATCHERS"

# ALL_SCHEDULER_GLOBAL_ATTRIBUTE

ALL_SCHEDULER_GLOBAL_ATTRIBUTE displays the values of all scheduler attributes (for example, DEFAULT_TIMEZONE and CURRENT_OPEN_WINDOW).

### Related View

DBA_SCHEDULER_GLOBAL_ATTRIBUTE displays the values of all scheduler attributes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ATTRIBUTE_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler attribute |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| VALUE | VARCHAR2(128) | | Value of the Scheduler attribute |

**See Also:**

"DBA_SCHEDULER_GLOBAL_ATTRIBUTE"

# ALL_SCHEDULER_GROUP_MEMBERS

ALL_SCHEDULER_GROUP_MEMBERS displays information about the members of the Scheduler object groups accessible to the current user.

### Related Views

- DBA_SCHEDULER_GROUP_MEMBERS displays information about the members of all Scheduler object groups in the database.
- USER_SCHEDULER_GROUP_MEMBERS displays information about the members of the Scheduler object groups owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the group |
| GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the group |
| MEMBER_NAME | VARCHAR2(523) | | Name of the member of this group |

**See Also:**

- "DBA_SCHEDULER_GROUP_MEMBERS"
- "USER_SCHEDULER_GROUP_MEMBERS"

# ALL_SCHEDULER_GROUPS

ALL_SCHEDULER_GROUPS displays information about the Scheduler object groups accessible to the current user.

### Related Views

- DBA_SCHEDULER_GROUPS displays information about all Scheduler object groups in the database.
- USER_SCHEDULER_GROUPS displays information about the Scheduler object groups owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the group |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the group |
| GROUP_TYPE | VARCHAR2(13) | | Type of object contained in the group:<br>• WINDOW<br>• JOB<br>• DB_DEST<br>• EXTERNAL_DEST |
| ENABLED | VARCHAR2(5) | | Indicates whether the group is enabled (TRUE) or disabled (FALSE) |
| NUMBER_OF_MEMBERS | NUMBER | | Number of members in this group |
| COMMENTS | VARCHAR2(240) | | An optional comment about this group |

**See Also:**

- "DBA_SCHEDULER_GROUPS"
- "USER_SCHEDULER_GROUPS"

# ALL_SCHEDULER_JOB_ARGS

ALL_SCHEDULER_JOB_ARGS displays information about the arguments of the Scheduler jobs accessible to the current user.

### Related Views

- DBA_SCHEDULER_JOB_ARGS displays information about the arguments of all Scheduler jobs in the database.
- USER_SCHEDULER_JOB_ARGS displays information about the arguments of the Scheduler jobs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the job to which the argument belongs |
| JOB_NAME | VARCHAR2(128) | | Name of the job to which the argument belongs |
| ARGUMENT_NAME | VARCHAR2(128) | | Optional name of the argument |
| ARGUMENT_POSITION | NUMBER | | Position of the argument in the argument list |
| ARGUMENT_TYPE | VARCHAR2(257) | | Data type of the argument |
| VALUE | VARCHAR2(4000) | | Value of the argument (in string format) if the argument is a string |
| ANYDATA_VALUE | ANYDATA | | Value of the argument (in AnyData format) |
| OUT_ARGUMENT | VARCHAR2(5) | | Reserved for future use |

**See Also:**

- "DBA_SCHEDULER_JOB_ARGS"
- "USER_SCHEDULER_JOB_ARGS"

# ALL_SCHEDULER_JOB_CLASSES

ALL_SCHEDULER_JOB_CLASSES displays information about the Scheduler job classes accessible to the current user.

### Related View

DBA_SCHEDULER_JOB_CLASSES displays information about all Scheduler job classes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler job class |
| JOB_CLASS_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler job class |
| RESOURCE_CONSUMER_GROUP | VARCHAR2(128) | | Resource consumer group associated with the class |
| SERVICE | VARCHAR2(64) | | Name of the service the class is associated with |
| LOGGING_LEVEL | VARCHAR2(11) | | Amount of logging that will be done pertaining to the class:<br>• OFF<br>• RUNS<br>• FAILED RUNS<br>• FULL |
| LOG_HISTORY | NUMBER | | History (in days) to maintain in the job log for the class |
| COMMENTS | VARCHAR2(240) | | Comments on the class |

**See Also:**

"DBA_SCHEDULER_JOB_CLASSES"

# ALL_SCHEDULER_JOB_DESTS

ALL_SCHEDULER_JOB_DESTS displays information about the state of the jobs accessible to the current user at each of their destinations.

### Related Views

- DBA_SCHEDULER_JOB_DESTS displays information about the state of all jobs in the database at each of their destinations.

- USER_SCHEDULER_JOB_DESTS displays information about the state of the jobs owned by the current user at each of their destinations. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the Scheduler job |
| JOB_NAME | VARCHAR2(128) | | Name of the Scheduler job |
| JOB_SUBNAME | VARCHAR2(128) | | Subname of the Scheduler job |
| CREDENTIAL_OWNER | VARCHAR2(128) | | Owner of the credential used for the remote destination |
| CREDENTIAL_NAME | VARCHAR2(128) | | Name of the credential used for the remote destination |
| DESTINATION_OWNER | VARCHAR2(128) | | Owner of the destination object that points to the destination |
| DESTINATION | VARCHAR2(128) | | Name of the destination object or the name of the destination itself |
| JOB_DEST_ID | NUMBER | | Numerical ID assigned to the job at this destination |
| ENABLED | VARCHAR2(5) | | Indicates whether the parent job is enabled (TRUE) or disabled (FALSE) |
| REFS_ENABLED | VARCHAR2(5) | | Indicates whether this destination and its agent are enabled (TRUE) or disabled (FALSE) |
| STATE | VARCHAR2(15) | | State of this job at this destination:<br>• DISABLED<br>• RUNNING<br>• CHAIN_STALLED<br>• SCHEDULED<br>• RETRY SCHEDULED<br>• READY TO RUN<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• SUCCEEDED<br>• REMOTE<br>• STOPPED |
| NEXT_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Next start time of this job at this destination |
| RUN_COUNT | NUMBER | | Number of times this job has run at this destination |
| RETRY_COUNT | NUMBER | | Number of times this job has been retried at this destination |
| FAILURE_COUNT | NUMBER | | Number of times this job has failed at this destination |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Last time this job started at this destination |
| LAST_END_DATE | TIMESTAMP(6) WITH TIME ZONE | | Last time this job ended at this destination |

**See Also:**

- "DBA_SCHEDULER_JOB_DESTS"
- "USER_SCHEDULER_JOB_DESTS"

# ALL_SCHEDULER_JOB_LOG

ALL_SCHEDULER_JOB_LOG displays log information for the Scheduler jobs accessible to the current user.

### Related Views

- DBA_SCHEDULER_JOB_LOG displays log information for all Scheduler jobs in the database.
- USER_SCHEDULER_JOB_LOG displays log information for the Scheduler jobs owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOG_ID | NUMBER | NOT NULL | Unique identifier that identifies a row |
| LOG_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date of the log entry |
| OWNER | VARCHAR2(128) | | Owner of the Scheduler job |
| JOB_NAME | VARCHAR2(261) | | Name of the Scheduler job |
| JOB_SUBNAME | VARCHAR2(261) | | Subname of the Scheduler job (for a chain step job) |
| JOB_CLASS | VARCHAR2(128) | | Class that the job belonged to at the time of entry |
| OPERATION | VARCHAR2(30) | | Operation corresponding to the log entry |
| STATUS | VARCHAR2(30) | | Status of the operation, if applicable. Possible values for this column are dependent on the value in the OPERATION column. In most cases, STATUS will be NULL. Only for job run operations will it have a value.<br>STATUS will be NULL when OPERATION is one of the following:<br>• CREATE - Job was created |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • UPDATE - One or more job attributes have been modified<br>• ENABLE - Job has been enabled<br>• DISABLE - Job has been disabled<br>• COMPLETED - For repeating jobs only, job has reached its end date or maximum number of runs<br>• BROKEN - Job has reached its maximum number of failures<br>STATUS can be SUCCEEDED (job run completed successfully), FAILED (job run failed), or STOPPED (job run was stopped) when OPERATION is one of the following:<br>• RUN - Regular job run<br>• RETRY_RUN - Job is being retried because the previous run resulted in an error and RESTARTABLE is set to TRUE<br>• RECOVERY_RUN - Job is being rerun because the database went down, or the job slave crashed and RESTARTABLE is set to TRUE |
| USER_NAME | VARCHAR2(128) | | Name of the user who performed the operation, if applicable |
| CLIENT_ID | VARCHAR2(64) | | Client identifier of the user who performed the operation, if applicable |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier of the user who performed the operation, if applicable |
| CREDENTIAL_OWNER | VARCHAR2(261) | | Owner of the credential used for this remote job run |
| CREDENTIAL_NAME | VARCHAR2(261) | | Name of the credential used for this remote job run |
| DESTINATION_OWNER | VARCHAR2(128) | | Owner of the destination object used in this remote job run; NULL if no object used |
| DESTINATION | VARCHAR2(128) | | Destination for a remote job operation |
| ADDITIONAL_INFO | CLOB | | Additional information on the entry, if applicable |

**See Also:**

- "DBA_SCHEDULER_JOB_LOG"
- "USER_SCHEDULER_JOB_LOG"

# ALL_SCHEDULER_JOB_RUN_DETAILS

ALL_SCHEDULER_JOB_RUN_DETAILS displays log run details for the Scheduler jobs accessible to the current user.

### Related Views

- DBA_SCHEDULER_JOB_RUN_DETAILS displays log run details for all Scheduler jobs in the database.
- USER_SCHEDULER_JOB_RUN_DETAILS displays log run details for the Scheduler jobs owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOG_ID | NUMBER | NOT NULL | Unique identifier of the log entry (foreign key of the *_SCHEDULER_JOB_LOG views) |
| LOG_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date of the log entry |
| OWNER | VARCHAR2(128) | | Owner of the Scheduler job |
| JOB_NAME | VARCHAR2(261) | | Name of the Scheduler job |
| JOB_SUBNAME | VARCHAR2(261) | | Subname of the Scheduler job (for a chain step job) |
| STATUS | VARCHAR2(30) | | Status of the job run |
| ERROR# | NUMBER | | Error number in the case of an error |
| REQ_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Requested start date of the job run |
| ACTUAL_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Actual date on which the job was run |
| RUN_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Duration of the job run |
| INSTANCE_ID | NUMBER | | Identifier of the instance on which the job was run |
| SESSION_ID | VARCHAR2(128) | | Session identifier of the job run |
| SLAVE_PID | VARCHAR2(30) | | Process identifier of the slave on which the job was run |
| CPU_USED | INTERVAL DAY(3) TO SECOND(2) | | Amount of CPU used for the job run |
| CREDENTIAL_OWNER | VARCHAR2(261) | | Owner of the credential used for this remote job run |
| CREDENTIAL_NAME | VARCHAR2(261) | | Name of the credential used for this remote job run |
| DESTINATION_OWNER | VARCHAR2(128) | | Owner of the destination object used in this remote job run; NULL if no object used |
| DESTINATION | VARCHAR2(128) | | Destination for a remote job operation |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADDITIONAL_INFO | VARCHAR2(4000) | | Additional information on the job run, if applicable |
| ERRORS | VARCHAR2(4000) | | Error messages generated by this job run |
| OUTPUT | VARCHAR2(4000) | | Output messages generated by this job run |
| BINARY_ERRORS | BLOB | | Error messages generated by this job run in a binary format |
| BINARY_OUTPUT | BLOB | | Binary output messages generated by this job run |

**See Also:**

- "DBA_SCHEDULER_JOB_RUN_DETAILS"
- "USER_SCHEDULER_JOB_RUN_DETAILS"

# ALL_SCHEDULER_JOBS

ALL_SCHEDULER_JOBS displays information about the Scheduler jobs accessible to the current user.

### Related Views

- DBA_SCHEDULER_JOBS displays information about all Scheduler jobs in the database.
- USER_SCHEDULER_JOBS displays information about the Scheduler jobs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the Scheduler job |
| JOB_NAME | VARCHAR2(128) | | Name of the Scheduler job |
| JOB_SUBNAME | VARCHAR2(128) | | Subname of the Scheduler job (for a job running a chain step) |
| JOB_STYLE | VARCHAR2(17) | | Job style:<br>• REGULAR<br>• LIGHTWEIGHT |
| JOB_CREATOR | VARCHAR2(128) | | Original creator of the job |
| CLIENT_ID | VARCHAR2(65) | | Client identifier of the user creating the job |
| GLOBAL_UID | VARCHAR2(33) | | Global user identifier of the user creating the job |
| PROGRAM_OWNER | VARCHAR2(4000) | | Owner of the program associated with the job |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROGRAM_NAME | VARCHAR2(4000) | | Name of the program associated with the job |
| JOB_TYPE | VARCHAR2(16) | | Inline job action type:<br>• PLSQL_BLOCK<br>• STORED_PROCEDURE<br>• EXECUTABLE<br>• CHAIN<br>• SQL_SCRIPT<br>• BACKUP_SCRIPT<br>• EXTERNAL_SCRIPT |
| JOB_ACTION | VARCHAR2(4000) | | Inline job action |
| NUMBER_OF_ARGUMENTS | NUMBER | | Inline number of job arguments |
| SCHEDULE_OWNER | VARCHAR2(4000) | | Owner of the schedule that the job uses (can be a window or a window group) |
| SCHEDULE_NAME | VARCHAR2(4000) | | Name of the schedule that the job uses (can be a window or a window group) |
| SCHEDULE_TYPE | VARCHAR2(12) | | Type of the schedule that the job uses:<br>• IMMEDIATE - Start date and repeat interval are NULL<br>• ONCE - Repeat interval is NULL<br>• PLSQL - PL/SQL expression used as schedule<br>• CALENDAR - Oracle calendaring expression used as schedule<br>• EVENT - Event schedule<br>• NAMED - Named schedule<br>• WINDOW - Window used as schedule<br>• WINDOW_GROUP - Window group used as schedule |
| START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Original scheduled start date of the job (for an inline schedule) |
| REPEAT_INTERVAL | VARCHAR2(4000) | | Inline schedule PL/SQL expression or calendar string |
| EVENT_QUEUE_OWNER | VARCHAR2(128) | | Owner of the source queue into which the event will be raised |
| EVENT_QUEUE_NAME | VARCHAR2(128) | | Name of the source queue into which the event will be raised |
| EVENT_QUEUE_AGENT | VARCHAR2(256) | | Name of the AQ agent used by the user on the event source queue (if it is a secure queue) |
| EVENT_CONDITION | VARCHAR2(4000) | | Boolean expression used as the subscription rule for the event on the source queue |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVENT_RULE | VARCHAR2(261) | | Name of the rule used by the coordinator to trigger the event-based job |
| FILE_WATCHER_OWNER | VARCHAR2(261) | | Owner of the file watcher on which this job is based |
| FILE_WATCHER_NAME | VARCHAR2(261) | | Name of the file watcher on which this job is based |
| END_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date after which the job will no longer run (for an inline schedule) |
| JOB_CLASS | VARCHAR2(128) | | Name of the job class associated with the job |
| ENABLED | VARCHAR2(5) | | Indicates whether the job is enabled (TRUE) or disabled (FALSE) |
| AUTO_DROP | VARCHAR2(5) | | Indicates whether the job will be dropped when it has completed (TRUE) or not (FALSE) |
| RESTART_ON_RECOVERY | VARCHAR2(5) | | Indicates whether the step should be restarted on database recovery (TRUE) or not (FALSE) |
| RESTART_ON_FAILURE | VARCHAR2(5) | | Indicates whether the step should be restarted on application failure (TRUE) or not (FALSE) |
| STATE | VARCHAR2(15) | | Current state of the job:<br>• DISABLED<br>• RETRY SCHEDULED<br>• SCHEDULED<br>• RUNNING<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• REMOTE<br>• SUCCEEDED<br>• CHAIN_STALLED |
| JOB_PRIORITY | NUMBER | | Priority of the job relative to other jobs in the same class |
| RUN_COUNT | NUMBER | | Number of times the job has run |
| MAX_RUNS | NUMBER | | Maximum number of times the job is scheduled to run |
| FAILURE_COUNT | NUMBER | | Number of times the job has failed to run |
| MAX_FAILURES | NUMBER | | Number of times the job will be allowed to fail before being marked broken |
| RETRY_COUNT | NUMBER | | Number of times the job has retried, if it is retrying |
| LAST_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Last date on which the job started running |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_RUN_DURATION | INTERVAL DAY(9) TO SECOND(6) | | Amount of time the job took to complete during the last run |
| NEXT_RUN_DATE | TIMESTAMP(6) WITH TIME ZONE | | Next date on which the job is scheduled to run |
| SCHEDULE_LIMIT | INTERVAL DAY(3) TO SECOND(0) | | Time after which a job which has not run yet will be rescheduled |
| MAX_RUN_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Maximum amount of time for which the job will be allowed to run |
| LOGGING_LEVEL | VARCHAR2(11) | | Amount of logging that will be done pertaining to the job:<br>• OFF<br>• RUNS<br>• FAILED RUNS<br>• FULL |
| STORE_OUTPUT | VARCHAR2(5) | | Indicates whether all job output messages for the job are stored in the OUTPUT column of the *_JOB_RUN_DETAILS views for job runs that are logged. Possible values:<br>• TRUE: All job output messages for the job are stored in the OUTPUT column of the *_JOB_RUN_DETAILS views for job runs that are logged. This is the default for new jobs. A new job is a job created using Oracle Database 12*c* software.<br>• FALSE: Job output messages for the job are not stored in the OUTPUT column of the *_JOB_RUN_DETAILS views. This is the default for existing jobs. An existing job is a job created using pre-Oracle Database 12*c* software. |
| STOP_ON_WINDOW_CLOSE | VARCHAR2(5) | | Indicates whether the job will stop if a window associated with the job closes (TRUE) or not (FALSE) |
| INSTANCE_STICKINESS | VARCHAR2(5) | | Indicates whether the job is sticky (TRUE) or not (FALSE) |
| RAISE_EVENTS | VARCHAR2(4000) | | List of job events to raise for the job:<br>• JOB_STARTED<br>• JOB_SUCCEEDED<br>• JOB_FAILED<br>• JOB_BROKEN<br>• JOB_COMPLETED<br>• JOB_STOPPED<br>• JOB_SCH_LIM_REACHED<br>• JOB_DISABLED<br>• JOB_CHAIN_STALLED |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • JOB_OVER_MAX_DUR |
| SYSTEM | VARCHAR2(5) | | Indicates whether the job is a system job (TRUE) or not (FALSE) |
| JOB_WEIGHT | NUMBER | | Weight of the job |
| NLS_ENV | VARCHAR2(4000) | | NLS environment of the job |
| SOURCE | VARCHAR2(128) | | Source global database identifier |
| NUMBER_OF_DESTINATIONS | NUMBER | | Number of destinations associated with this job |
| DESTINATION_OWNER | VARCHAR2(128) | | Owner of the destination object (if used), else NULL |
| DESTINATION | VARCHAR2(128) | | Destination that this job will run on |
| CREDENTIAL_OWNER | VARCHAR2(128) | | Owner of the credential to be used for an external job |
| CREDENTIAL_NAME | VARCHAR2(128) | | Name of the credential to be used for an external job |
| INSTANCE_ID | NUMBER | | Instance on which the user requests the job to run |
| DEFERRED_DROP | VARCHAR2(5) | | Indicates whether the job will be dropped when completed due to user request (TRUE) or not (FALSE) |
| ALLOW_RUNS_IN_RESTRICTED_MODE | VARCHAR2(5) | | Indicates whether the job is allowed to run in restricted session mode (TRUE) or not (FALSE) |
| COMMENTS | VARCHAR2(240) | | Comments on the job |
| FLAGS | NUMBER | | This column is for internal use |
| RESTARTABLE | VARCHAR2(5) | | Indicates whether the job can be restarted (TRUE) or not (FALSE) |
| CONNECT_CREDENTIAL_OWNER | VARCHAR2(128) | | Owner of connect credential |
| CONNECT_CREDENTIAL_NAME | VARCHAR2(128) | | Name of connect credential |

**See Also:**

- "DBA_SCHEDULER_JOBS"
- "USER_SCHEDULER_JOBS"

# ALL_SCHEDULER_NOTIFICATIONS

ALL_SCHEDULER_NOTIFICATIONS displays information about the E-mail notifications for the jobs accessible to the current user.

### Related Views

- DBA_SCHEDULER_NOTIFICATIONS displays information about the E-mail notifications for all jobs in the database.
- USER_SCHEDULER_NOTIFICATIONS displays information about the E-mail notifications for the jobs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the job this notification is for |
| JOB_NAME | VARCHAR2(128) | NOT NULL | Name of the job this notification is for |
| JOB_SUBNAME | VARCHAR2(128) | | Subname of the job this notification is for |
| RECIPIENT | VARCHAR2(4000) | NOT NULL | E-mail address to send this E-mail notification to |
| SENDER | VARCHAR2(4000) | | E-mail address to send this E-mail notification from |
| SUBJECT | VARCHAR2(4000) | | Subject of the notification E-mail |
| BODY | VARCHAR2(4000) | | Body of the notification E-mail |
| FILTER_CONDITION | VARCHAR2(4000) | | Filter specifying which job events to send notifications for |
| EVENT | VARCHAR2(19) | | Job event to send notifications for:<br>• JOB_STARTED<br>• JOB_SUCCEEDED<br>• JOB_FAILED<br>• JOB_BROKEN<br>• JOB_COMPLETED<br>• JOB_STOPPED<br>• JOB_SCH_LIM_REACHED<br>• JOB_DISABLED<br>• JOB_CHAIN_STALLED<br>• JOB_OVER_MAX_DUR |
| EVENT_FLAG | NUMBER | NOT NULL | Event number of the job event to send notifications for |

**See Also:**

- "DBA_SCHEDULER_NOTIFICATIONS"
- "USER_SCHEDULER_NOTIFICATIONS"

# ALL_SCHEDULER_PROGRAM_ARGS

ALL_SCHEDULER_PROGRAM_ARGS displays information about the arguments of the Scheduler programs accessible to the current user.

### Related Views

- DBA_SCHEDULER_PROGRAM_ARGS displays information about the arguments of all Scheduler programs in the database.
- USER_SCHEDULER_PROGRAM_ARGS displays information about the arguments of the Scheduler programs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the program to which the argument belongs |
| PROGRAM_NAME | VARCHAR2(128) | NOT NULL | Name of the program to which the argument belongs |
| ARGUMENT_NAME | VARCHAR2(128) | | Optional name of the argument |
| ARGUMENT_POSITION | NUMBER | NOT NULL | Position of the argument in the argument list |
| ARGUMENT_TYPE | VARCHAR2(257) | | Data type of the argument |
| METADATA_ATTRIBUTE | VARCHAR2(19) | | Metadata attribute:<br>• JOB_NAME<br>• JOB_OWNER<br>• JOB_START<br>• WINDOW_START<br>• WINDOW_END<br>• JOB_SUBNAME<br>• EVENT_MESSAGE<br>• JOB_SCHEDULED_START |
| DEFAULT_VALUE | VARCHAR2(4000) | | Default value taken by the argument (in string format) if the argument is a string |
| DEFAULT_ANYDATA_VALUE | ANYDATA | | Default value taken by the argument (in AnyData format) |
| OUT_ARGUMENT | VARCHAR2(5) | | Reserved for future use |

**See Also:**

- "DBA_SCHEDULER_PROGRAM_ARGS"
- "USER_SCHEDULER_PROGRAM_ARGS"

# ALL_SCHEDULER_PROGRAMS

ALL_SCHEDULER_PROGRAMS displays information about the Scheduler programs accessible to the current user.

### Related Views

- DBA_SCHEDULER_PROGRAMS displays information about all Scheduler programs in the database.
- USER_SCHEDULER_PROGRAMS displays information about the Scheduler programs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler program |
| PROGRAM_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler program |
| PROGRAM_TYPE | VARCHAR2(16) | | Type of the program action:<br>• PLSQL_BLOCK<br>• STORED_PROCEDURE<br>• EXECUTABLE |
| PROGRAM_ACTION | VARCHAR2(4000) | | String specifying the program action |
| NUMBER_OF_ARGUMENTS | NUMBER | | Number of arguments accepted by the program |
| ENABLED | VARCHAR2(5) | | Indicates whether the program is enabled (TRUE) or disabled (FALSE) |
| DETACHED | VARCHAR2(5) | | This column is for internal use |
| SCHEDULE_LIMIT | INTERVAL DAY(3) TO SECOND(0) | | Maximum delay in running the program after the scheduled start |
| PRIORITY | NUMBER | | Priority of the program |
| WEIGHT | NUMBER | | Weight of the program |
| MAX_RUNS | NUMBER | | Maximum number of runs of any job based on this program |
| MAX_FAILURES | NUMBER | | Maximum number of failures of any job based on this program |
| MAX_RUN_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Maximum amount of time this program can run |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NLS_ENV | VARCHAR2(4000) | | NLS environment in which the program was created |
| COMMENTS | VARCHAR2(240) | | Comments on the program |

**See Also:**

- "DBA_SCHEDULER_PROGRAMS"
- "USER_SCHEDULER_PROGRAMS"

# ALL_SCHEDULER_REMOTE_DATABASES

ALL_SCHEDULER_REMOTE_DATABASES displays information about the remote databases accessible to the current user that have been registered as sources and destinations for remote database jobs.

### Related View

DBA_SCHEDULER_REMOTE_DATABASES displays information about all remote databases that have been registered as sources and destinations for remote database jobs.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DATABASE_NAME | VARCHAR2(512) | NOT NULL | Global name of the remote database |
| REGISTERED_AS | VARCHAR2(11) | | Indicates whether the database is registered as a source (SOURCE) or as a destination (DESTINATION) |
| DATABASE_LINK | VARCHAR2(512) | NOT NULL | Name of a valid database link to the remote database |

**See Also:**

"DBA_SCHEDULER_REMOTE_DATABASES"

# ALL_SCHEDULER_REMOTE_JOBSTATE

ALL_SCHEDULER_REMOTE_JOBSTATE displays information about the state of the jobs accessible to the current user at remote databases.

### Related Views

- DBA_SCHEDULER_REMOTE_JOBSTATE displays information about the state of all jobs at remote databases.
- USER_SCHEDULER_REMOTE_JOBSTATE displays information about the state of the jobs owned by the current user at remote databases. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler job |
| JOB_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler job |
| DESTINATION | VARCHAR2(512) | NOT NULL | Name of the job destination |
| STATE | VARCHAR2(15) | | State of the job at the destination:<br>• DISABLED<br>• RETRY SCHEDULED<br>• SCHEDULED<br>• RUNNING<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• SUCCEEDED<br>• STOPPED |
| NEXT_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Next start date of the job at the destination |
| RUN_COUNT | NUMBER | | Run count of the job at the destination |
| FAILURE_COUNT | NUMBER | | Failure count of the job at the destination |
| RETRY_COUNT | NUMBER | | Retry count of the job at the destination |
| LAST_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Last start date of the job at the destination |
| LAST_END_DATE | TIMESTAMP(6) WITH TIME ZONE | | Last end date of the job at the destination |

**See Also:**

- "DBA_SCHEDULER_REMOTE_JOBSTATE"
- "USER_SCHEDULER_REMOTE_JOBSTATE"

# ALL_SCHEDULER_RUNNING_CHAINS

ALL_SCHEDULER_RUNNING_CHAINS displays information about the chain steps of the running chains accessible to the current user (that is, those chains that the user has ALTER privileges for). In the case of nested chains, this view also enables you to traverse the hierarchy of the chain with a SQL statement that contains a CONNECT BY clause linking up the JOB_SUBNAME and STEP_JOB_SUBNAME columns.

### Related Views

- DBA_SCHEDULER_RUNNING_CHAINS displays information about the chain steps of all running chains in the database.

- USER_SCHEDULER_RUNNING_CHAINS displays information about the chain steps of the running chains owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the job which is running the chain |
| JOB_NAME | VARCHAR2(128) | NOT NULL | Name of the job which is running the chain |
| JOB_SUBNAME | VARCHAR2(128) | | Subname of the job which is running the chain (for a nested chain), else NULL |
| CHAIN_OWNER | VARCHAR2(128) | NOT NULL | Owner of the chain being run |
| CHAIN_NAME | VARCHAR2(128) | NOT NULL | Name of the chain being run |
| STEP_NAME | VARCHAR2(128) | NOT NULL | Name of the step of the running chain |
| STATE | VARCHAR2(15) | | State of the running chain step:<br>• NOT_STARTED<br>• RUNNING<br>• SUCCEEDED<br>• STOPPED<br>• FAILED<br>• SCHEDULED<br>• RETRY SCHEDULED<br>• PAUSED<br>• STALLED |
| ERROR_CODE | NUMBER | | Error code with which the step completed (if it has completed) |
| COMPLETED | VARCHAR2(5) | | Indicates whether the running chain step has completed (TRUE) or not (FALSE) |
| START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date when the running chain step started (if it has started) |
| END_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date when the running chain step stopped (if it has stopped) |
| DURATION | INTERVAL DAY(9) TO SECOND(6) | | Amount of time it took the chain step to complete (if it has completed) |
| SKIP | VARCHAR2(5) | | Indicates whether the chain step should be skipped (TRUE) or not (FALSE) |
| PAUSE | VARCHAR2(5) | | Indicates whether the chain step should be paused after running (TRUE) or not (FALSE) |
| PAUSE_BEFORE | VARCHAR2(5) | | Indicates whether the chain step should be paused before running (TRUE) or not (FALSE) |
| RESTART_ON_RECOVERY | VARCHAR2(5) | | Indicates whether the chain step will be restarted on database recovery (TRUE) or not (FALSE) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RESTART_ON_FAILURE | VARCHAR2(5) | | Indicates whether the chain step will be restarted on application failure (TRUE) or not (FALSE) |
| STEP_JOB_SUBNAME | VARCHAR2(128) | | Subname of the job running the step |
| STEP_JOB_LOG_ID | NUMBER | | Log ID of the job running the step |

**See Also:**

- "DBA_SCHEDULER_RUNNING_CHAINS"
- "USER_SCHEDULER_RUNNING_CHAINS"

# ALL_SCHEDULER_RUNNING_JOBS

ALL_SCHEDULER_RUNNING_JOBS displays information about the running Scheduler jobs accessible to the current user.

### Related Views

- DBA_SCHEDULER_RUNNING_JOBS displays information about all running Scheduler jobs in the database.
- USER_SCHEDULER_RUNNING_JOBS displays information about the running Scheduler jobs owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the running Scheduler job |
| JOB_NAME | VARCHAR2(128) | | Name of the running Scheduler job |
| JOB_SUBNAME | VARCHAR2(128) | | Subname of the running Scheduler job (for a job running a chain step) |
| JOB_STYLE | VARCHAR2(17) | | Job style:<br>• REGULAR<br>• LIGHTWEIGHT |
| DETACHED | VARCHAR2(5) | | Indicates whether the detached attribute is set for the job (TRUE) or not (FALSE). If the detached attribute is set, then the job will remain running even after the job action has completed. |
| SESSION_ID | NUMBER | | Identifier of the session running the Scheduler job |
| SLAVE_PROCESS_ID | NUMBER | | Process number of the slave process running the Scheduler job |
| SLAVE_OS_PROCESS_ID | VARCHAR2(12) | | Process number of the operating system slave process running the scheduler job |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RUNNING_INSTANCE | NUMBER | | Database instance number of the slave process running the Scheduler job |
| RESOURCE_CONSUMER_GROUP | VARCHAR2(32) | | Resource consumer group of the session in which the Scheduler job is running |
| ELAPSED_TIME | INTERVAL DAY(3) TO SECOND(2) | | Elapsed time since the Scheduler job was started |
| CPU_USED | INTERVAL DAY(3) TO SECOND(2) | | CPU time consumed by the running Scheduler job, if available |
| DESTINATION_OWNER | VARCHAR2(128) | | Owner of the destination object (if used), else NULL |
| DESTINATION | VARCHAR2(128) | | Destination that this job is running on |
| CREDENTIAL_OWNER | VARCHAR2(128) | | Owner of the login credential used for this running job, if any |
| CREDENTIAL_NAME | VARCHAR2(128) | | Name of the login credential used for this running job, if any |
| LOG_ID | NUMBER | | Log ID used for this running job. This column maps to the LOG_ID column of the *_SCHEDULER_JOB_LOG and *_SCHEDULER_JOB_RUN_DETAILS views. |

**See Also:**

- "DBA_SCHEDULER_RUNNING_JOBS"
- "USER_SCHEDULER_RUNNING_JOBS"

# ALL_SCHEDULER_SCHEDULES

ALL_SCHEDULER_SCHEDULES displays information about the Scheduler schedules accessible to the current user.

### Related Views

- DBA_SCHEDULER_SCHEDULES displays information about all Scheduler schedules in the database.
- USER_SCHEDULER_SCHEDULES displays information about the Scheduler schedules owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the schedule |
| SCHEDULE_NAME | VARCHAR2(128) | NOT NULL | Name of the schedule |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEDULE_TYPE | VARCHAR2(12) | | Type of the schedule:<br>• ONCE - Repeat interval is NULL<br>• CALENDAR - Oracle calendaring expression used as schedule<br>• EVENT - Event schedule |
| START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Start date for the repeat interval |
| REPEAT_INTERVAL | VARCHAR2(4000) | | Calendar syntax expression for the schedule |
| EVENT_QUEUE_OWNER | VARCHAR2(128) | | Owner of the source queue into which the event will be raised |
| EVENT_QUEUE_NAME | VARCHAR2(128) | | Name of the source queue into which the event will be raised |
| EVENT_QUEUE_AGENT | VARCHAR2(128) | | Name of the AQ agent used by the user on the event source queue (if it is a secure queue) |
| EVENT_CONDITION | VARCHAR2(4000) | | Boolean expression used as the subscription rule for the event on the source queue |
| FILE_WATCHER_OWNER | VARCHAR2(261) | | Owner of the file watcher on which this schedule is based |
| FILE_WATCHER_NAME | VARCHAR2(261) | | Name of the file watcher on which this schedule is based |
| END_DATE | TIMESTAMP(6) WITH TIME ZONE | | Cutoff date after which the schedule will not specify any dates |
| COMMENTS | VARCHAR2(240) | | Comments on the schedule |

**See Also:**

- "DBA_SCHEDULER_SCHEDULES"
- "USER_SCHEDULER_SCHEDULES"

# ALL_SCHEDULER_WINDOW_DETAILS

ALL_SCHEDULER_WINDOW_DETAILS displays log details for the Scheduler windows accessible to the current user.

### Related View

DBA_SCHEDULER_WINDOW_DETAILS displays log details for all Scheduler windows in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOG_ID | NUMBER | | Unique identifier of the log entry (foreign key of the *_SCHEDULER_WINDOW_LOG views) |
| LOG_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date of the log entry |
| OWNER | VARCHAR2(128) | | Owner of the Scheduler window |
| WINDOW_NAME | VARCHAR2(261) | | Name of the Scheduler window |
| REQ_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Requested start date for the Scheduler window |
| ACTUAL_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Actual start date of the Scheduler window |
| WINDOW_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Requested duration of the Scheduler window |
| ACTUAL_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Actual duration for which the Scheduler window lasted |
| INSTANCE_ID | NUMBER | | Identifier of the instance on which the window was run |
| ADDITIONAL_INFO | VARCHAR2(4000) | | Additional information on the entry, if applicable |

**See Also:**

"DBA_SCHEDULER_WINDOW_DETAILS"

# ALL_SCHEDULER_WINDOW_GROUPS

ALL_SCHEDULER_WINDOW_GROUPS displays information about the Scheduler window groups accessible to the current user.

### Related View

DBA_SCHEDULER_WINDOW_GROUPS displays information about all Scheduler window groups in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WINDOW_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the window group |
| ENABLED | VARCHAR2(5) | | Indicates whether the window group is enabled (TRUE) or disabled (FALSE) |
| NUMBER_OF_WINDOWS | NUMBER | | Number of members in the window group |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NEXT_START_DATE | VARCHAR2(64) | | If a window group is disabled, then this column will be NULL. Otherwise, it will be set to the earliest NEXT_START_DATE from the enabled windows in the group. |
| COMMENTS | VARCHAR2(240) | | Optional comment about the window group |

**See Also:**

"DBA_SCHEDULER_WINDOW_GROUPS"

# ALL_SCHEDULER_WINDOW_LOG

ALL_SCHEDULER_WINDOW_LOG displays log information for the Scheduler windows accessible to the current user.

### Related View

DBA_SCHEDULER_WINDOW_LOG displays log information for all Scheduler windows in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOG_ID | NUMBER | NOT NULL | Unique identifier of the log entry |
| LOG_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date of the log entry |
| OWNER | VARCHAR2(128) | | Owner of the Scheduler window |
| WINDOW_NAME | VARCHAR2(261) | | Name of the Scheduler window |
| OPERATION | VARCHAR2(30) | | Operation corresponding to the log entry |
| STATUS | VARCHAR2(30) | | Status of the operation, if applicable |
| USER_NAME | VARCHAR2(128) | | Name of the user who performed the operation, if applicable |
| CLIENT_ID | VARCHAR2(64) | | Client identifier of the user who performed the operation, if applicable |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier of the user who performed the operation, if applicable |
| ADDITIONAL_INFO | CLOB | | Additional information on the entry, if applicable |

**See Also:**

"DBA_SCHEDULER_WINDOW_LOG"

# ALL_SCHEDULER_WINDOWS

ALL_SCHEDULER_WINDOWS displays information about the Scheduler windows accessible to the current user.

### Related View

DBA_SCHEDULER_WINDOWS displays information about all Scheduler windows in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler window |
| WINDOW_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler window |
| RESOURCE_PLAN | VARCHAR2(128) | | Resource plan associated with the window |
| SCHEDULE_OWNER | VARCHAR2(4000) | | Owner of the schedule of the window |
| SCHEDULE_NAME | VARCHAR2(4000) | | Name of the schedule of the window |
| SCHEDULE_TYPE | VARCHAR2(8) | | Type of the schedule of the window:<br>• ONCE - Repeat interval is NULL<br>• NAMED - Named schedule<br>• CALENDAR - Oracle calendaring expression used as schedule |
| START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Start date of the window (for an inline schedule) |
| REPEAT_INTERVAL | VARCHAR2(4000) | | Calendar string for the window (for an inline schedule) |
| END_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date after which the window will no longer open (for an inline schedule) |
| DURATION | INTERVAL DAY(3) TO SECOND(0) | | Duration of the window |
| WINDOW_PRIORITY | VARCHAR2(4) | | Priority of the job relative to other windows:<br>• HIGH<br>• LOW |
| NEXT_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Next date on which the window is scheduled to start |
| LAST_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Last date on which the window opened |
| ENABLED | VARCHAR2(5) | | Indicates whether the window is enabled (TRUE) or disabled (FALSE) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ACTIVE | VARCHAR2(5) | | Indicates whether the window is open (TRUE) or not (FALSE) |
| MANUAL_OPEN_TIME | TIMESTAMP(6) WITH TIME ZONE | | Open time of the window if it was manually opened, else NULL |
| MANUAL_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Duration of the window if it was manually opened, else NULL |
| COMMENTS | VARCHAR2(240) | | Comments on the window |

**See Also:**

"DBA_SCHEDULER_WINDOWS"

# ALL_SCHEDULER_WINGROUP_MEMBERS

ALL_SCHEDULER_WINGROUP_MEMBERS displays the members of the Scheduler window groups accessible to the current user.

### Related View

DBA_SCHEDULER_WINGROUP_MEMBERS displays the members of all Scheduler window groups in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WINDOW_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the window group |
| WINDOW_NAME | VARCHAR2(128) | NOT NULL | Name of the window member of the window group |

**See Also:**

"DBA_SCHEDULER_WINGROUP_MEMBERS"

# ALL_SEC_RELEVANT_COLS

ALL_SEC_RELEVANT_COLS describes the security relevant columns of the security policies for the tables and views accessible to the current user.

### Related Views

- DBA_SEC_RELEVANT_COLS describes the security relevant columns of all security policies in the database.
- USER_SEC_RELEVANT_COLS describes the security relevant columns of the security policies for the tables and views owned by the current user. This view does not display the OBJECT_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the table or view |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or view |
| POLICY_GROUP | VARCHAR2(128) | | Name of the policy group |
| POLICY_NAME | VARCHAR2(128) | | Name of the policy |
| SEC_REL_COLUMN | VARCHAR2(128) | | Name of the security relevant column |
| COLUMN_OPTION | VARCHAR2(8) | | Option of the security relevant column:<br>• NONE<br>• ALL_ROWS |

**See Also:**

- "DBA_SEC_RELEVANT_COLS"
- "USER_SEC_RELEVANT_COLS"

# ALL_SECONDARY_OBJECTS

ALL_SECONDARY_OBJECTS provides information about secondary objects associated with domain indexes accessible to the user. This view is only relevant for domain indexes. And currently, the secondary objects can only be tables.

### Related Views

- DBA_SECONDARY_OBJECTS provides information about all secondary objects that are associated with domain indexes in the database.
- USER_SECONDARY_OBJECTS provides information about secondary objects associated with domain indexes owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_OWNER | VARCHAR2(128) | NOT NULL | Owner of the domain index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the domain index |
| SECONDARY_OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the secondary object created by the domain index |
| SECONDARY_OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the secondary object created by the domain index |
| SECONDARY_OBJDATA_TYPE | VARCHAR2(20) | | Type of the secondary object created by the domain index |

**See Also:**

- "DBA_SECONDARY_OBJECTS"
- "USER_SECONDARY_OBJECTS"

# ALL_SEQUENCES

ALL_SEQUENCES describes all sequences accessible to the current user.

### Related Views

- DBA_SEQUENCES describes all sequences in the database.
- USER_SEQUENCES describes all sequences owned by the current user. This view does not display the SEQUENCE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEQUENCE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the sequence |
| SEQUENCE_NAME | VARCHAR2(128) | NOT NULL | Sequence name |
| MIN_VALUE | NUMBER | | Minimum value of the sequence |
| MAX_VALUE | NUMBER | | Maximum value of the sequence |
| INCREMENT_BY | NUMBER | NOT NULL | Value by which sequence is incremented |
| CYCLE_FLAG | VARCHAR2(1) | | Indicates whether the sequence wraps around on reaching the limit (Y) or not (N) |
| ORDER_FLAG | VARCHAR2(1) | | Indicates whether sequence numbers are generated in order (Y) or not (N) |
| CACHE_SIZE | NUMBER | NOT NULL | Number of sequence numbers to cache |
| LAST_NUMBER | NUMBER | NOT NULL | Last sequence number written to disk. If a sequence uses caching, the number written to disk is the last number placed in the sequence cache. This number is likely to be greater than the last sequence number that was used.<br>For session sequences, the value in this column should be ignored. |
| PARTITION_COUNT | NUMBER | | Number of partitions in the sequence |
| SESSION_FLAG | VARCHAR2(1) | | Indicates whether sequence values are session private (Y) or not (N). |
| KEEP_VALUE | VARCHAR2(1) | | Indicates whether sequence values are kept during reply after a failure (Y) or not (N) |

**See Also:**

- "DBA_SEQUENCES"
- "USER_SEQUENCES"

# ALL_SERVICES

ALL_SERVICES displays all services in the database. The view excludes rows marked for deletion.

### Related View

DBA_SERVICES displays all services in the database. The view excludes rows marked for deletion.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SERVICE_ID | NUMBER | | Unique ID for the service |
| NAME | VARCHAR2(64) | | Name describing the workload |
| NAME_HASH | NUMBER | | Hash of the short name for the service |
| NETWORK_NAME | VARCHAR2(512) | | Network name used to connect to the service |
| CREATION_DATE | DATE | | Date the service was created |
| CREATION_DATE_HASH | NUMBER | | Hash of the creation date |
| FAILOVER_METHOD | VARCHAR2(64) | | TAF only for compatibility - BASIC or NONE |
| FAILOVER_TYPE | VARCHAR2(64) | | For Application Continuity for Java, TRANSACTION. For TAF, SESSION or SELECT. |
| FAILOVER_RETRIES | NUMBER(10) | | For Application Continuity and TAF, when reconnecting after a failure, number of attempts to re-connect per incident |
| FAILOVER_DELAY | NUMBER(10) | | For Application Continuity and TAF, when reconnecting after a failure, delay between each connection retry (in seconds) |
| MIN_CARDINALITY | NUMBER | | Reserved for internal use |
| MAX_CARDINALITY | NUMBER | | Reserved for internal use |
| GOAL | VARCHAR2(12) | | Runtime Load Balancing Goal being used to create run-time load balancing and connection load balancing advice:<br>• NONE<br>• SERVICE_TIME - Connections are balanced by response time<br>• THROUGHPUT - Connections are balanced by throughput |
| DTP | VARCHAR2(1) | | DTP (distributed transaction processing) enforces all sessions for a service at one instance. This is a requirement for XA before 11*g*, and is a |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | requirement if resuming and suspending the same XA branch. |
| ENABLED | VARCHAR2(3) | | Reserved for internal use |
| AQ_HA_NOTIFICATIONS | VARCHAR2(3) | | To enable FAN for OCI connections, set AQ HA Notifications to `True`. For Oracle Database 12*c*, FAN uses ONS (Oracle Notification Service) |
| CLB_GOAL | VARCHAR2(5) | | Connection load balancing goal. When using run-time load balancing, `GOAL=SERVICE_TIME` or `THROUGHPUT`, set to `SHORT`. For a session count per service only, use `LONG`. |
| EDITION | VARCHAR2(128) | | A non-NULL value specifies the initial session edition for subsequent database connections that use the service and do not specify an edition. A NULL value has no effect. |
| COMMIT_OUTCOME | VARCHAR2(3) | | This attribute is used for Transaction Guard. When `COMMIT_OUTCOME` is set to `TRUE`, Transaction Guard manages the commit status for all supported transaction types. The attribute is on a per session basis and is set at connect time.<br>When the attribute is enabled:<br>• The outcome of a COMMIT transaction is known. If there is an outage, the application can use `DBMS_APP_CONT.GET_LTXID_OUTCOME` to return a reliable status for the last in-flight work.<br>• A logical transaction ID (LTXID) is set for each user session at login and at each successful commit.<br>**See Also:** For information about preserving the commit outcome, see *Oracle Database Development Guide*. For information about logical transaction IDs, see *Oracle Database Development Guide* |
| RETENTION_TIMEOUT | NUMBER | | For Transaction Guard (`COMMIT_OUTCOME` set to `TRUE`), this parameter determines the amount of time (in seconds) that the commit outcome is retained in the database |
| REPLAY_INITIATION_TIMEOUT | NUMBER | | For Application Continuity, indicates a time period (in seconds) after which the request will not be replayed. The time period starts at the first request submission.<br>The default value is 300 seconds, which is 5 minutes. |
| SESSION_STATE_CONSISTENCY | VARCHAR2(128) | | Describes how non-transactional is changed during a request. This parameter is considered only if `failover_type` is set to `TRANSACTION` for Application Continuity. Examples of session state are NLS settings, optimizer preferences, event settings, PL/SQL global variables, temporary tables, advanced queues, LOBs and |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | result cache. If non-transactional values change after the request starts, the default value of `DYNAMIC` should be set. Almost all applications should use `DYNAMIC` mode. If you are unsure, use `DYNAMIC` mode. |
| `GLOBAL_SERVICE` | `VARCHAR2(3)` | | Indicates whether the service is global. A global service is managed by Global Data Services (GDS) and can be provided by multiple databases that contain replicated data. Possible values:<br>• `YES`: Indicates the service is global<br>• `NO`: Indicates the service is not global<br>This attribute is set when using GDS. It cannot be set by a user. |
| `PDB` | `VARCHAR2(128)` | | Name of a PDB associated with a given service. Will contain NULL for a non-CDB or if the service is not associated with a PDB (that is, connecting to a CDB using this service will cause a user to connect to the root.)<br>When managing services for a PDB, use SRVCTL for Oracle RAC and Oracle RAC One Node databases, or connect to that PDB if it is a single instance (not RAC). The PDB attribute shows which PDB has the service. It cannot be set or modified explicitly. |
| `SQL_TRANSLATION_PROFILE` | `VARCHAR2(261)` | | A non-NULL value specifies the initial SQL translation profile for subsequent database connections that use the service and do not specify a SQL translation profile. A NULL value has no effect. |
| `MAX_LAG_TIME` | `VARCHAR2(128)` | | The maximum replication lag (in seconds) that is acceptable for a data replica to be used for providing the database service. Can only be specified for global services using the Global Data Services (GDS) interfaces. It is not supported to change this value at local databases. |
| `GSM_FLAGS` | `NUMBER` | | Flags specific to Global Data Services (GDS). Can only be specified for global services using the GDS interfaces. It is not supported to change these values at local databases. |

**See Also:**

- "DBA_SERVICES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_APP_CONT.GET_LTXID_OUTCOME` procedure

# ALL_SOURCE

ALL_SOURCE describes the text source of the stored objects accessible to the current user.

### Related Views

- DBA_SOURCE describes the text source of all stored objects in the database.
- USER_SOURCE describes the text source of the stored objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| TYPE | VARCHAR2(12) | | Type of object: FUNCTION, JAVA SOURCE, PACKAGE, PACKAGE BODY, PROCEDURE, TRIGGER, TYPE, TYPE BODY |
| LINE | NUMBER | NOT NULL | Line number of this line of source |
| TEXT | VARCHAR2(4000) | | Text source of the stored object |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_SOURCE"
- "USER_SOURCE"

# ALL_SOURCE_AE

ALL_SOURCE_AE describes the text source of the stored objects (across all editions) accessible to the current user.

### Related Views

- DBA_SOURCE_AE describes the text source of all stored objects (across all editions) in the database.
- USER_SOURCE_AE describes the text source of the stored objects (across all editions) owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the object |
| NAME | VARCHAR2(128) | | Name of the object |
| TYPE | VARCHAR2(12) | | Type of the object:<br>• TYPE<br>• TYPE BODY<br>• PROCEDURE<br>• FUNCTION<br>• PACKAGE<br>• PACKAGE BODY<br>• LIBRARY<br>• JAVA SOURCE |
| LINE | NUMBER | | Line number of this line of source |
| TEXT | VARCHAR2(4000) | | Source text |
| EDITION_NAME | VARCHAR2(128) | | Name of the Edition |

**See Also:**

- "DBA_SOURCE_AE"
- "USER_SOURCE_AE"

# ALL_SQL_TRANSLATION_PROFILES

ALL_SQL_TRANSLATION_PROFILES describes all SQL translation profiles accessible to the user.

### Related Views

- DBA_SQL_TRANSLATION_PROFILES describes all SQL translation profiles in the database.
- USER_SQL_TRANSLATION_PROFILES describes all SQL translation profiles owned by the user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the SQL translation profile |
| PROFILE_NAME | VARCHAR2(128) | NOT NULL | Name of the SQL translation profile |
| TRANSLATOR | VARCHAR2(261) | | The translator package |
| FOREIGN_SQL_SYNTAX | VARCHAR2(5) | | Indicates whether the SQL syntax is foreign.<br>Possible values:<br>• TRUE<br>• FALSE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRANSLATE_NEW_SQL | VARCHAR2(5) | | Indicates whether to translate new SQL statements and errors using the translator.<br>Possible values:<br>• TRUE<br>• FALSE |
| RAISE_TRANSLATION_ERROR | VARCHAR2(5) | | Indicates whether to raise translation error.<br>Possible values:<br>• TRUE<br>• FALSE |
| LOG_TRANSLATION_ERROR | VARCHAR2(5) | | Indicates whether to log translation error.<br>Possible values:<br>• TRUE<br>• FALSE |
| TRACE_TRANSLATION | VARCHAR2(5) | | Indicates whether to trace translation.<br>Possible values:<br>• TRUE<br>• FALSE |

**See Also:**

- "DBA_SQL_TRANSLATION_PROFILES"
- "USER_SQL_TRANSLATION_PROFILES"

# ALL_SQL_TRANSLATIONS

ALL_SQL_TRANSLATIONS describes all SQL translations accessible to the user.

### Related Views

- DBA_SQL_TRANSLATIONS describes all SQL translations in the database.
- USER_SQL_TRANSLATIONS describes all SQL translations owned by the user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the SQL translation profile |
| PROFILE_NAME | VARCHAR2(128) | NOT NULL | Name of the SQL translation profile |
| SQL_TEXT | CLOB | NOT NULL | The SQL statement |
| TRANSLATED_TEXT | CLOB | | The translated SQL statement |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the SQL statement |
| HASH_VALUE | NUMBER | NOT NULL | Hash value of the SQL statement |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ENABLED | VARCHAR2(5) | | Displays whether the translation is enabled. Possible values:<br>• TRUE<br>• FALSE |
| REGISTRATION_TIME | TIMESTAMP(6) | | Time the translation was registered |
| CLIENT_INFO | VARCHAR2(64) | | Client information when the SQL was parsed and the translation was registered |
| MODULE | VARCHAR2(64) | | Module when the SQL was parsed and the translation was registered |
| ACTION | VARCHAR2(64) | | Action when the SQL was parsed and the translation was registered |
| PARSING_USER_ID | NUMBER | | Current user ID when the SQL was parsed and the translation was registered |
| PARSING_SCHEMA_ID | NUMBER | | Current schema ID when the SQL was parsed and the translation was registered |
| COMMENTS | VARCHAR2(4000) | | Comment on the translation |

**See Also:**

- "DBA_SQL_TRANSLATIONS"
- "USER_SQL_TRANSLATIONS"

# ALL_SQLJ_TYPE_ATTRS

ALL_SQLJ_TYPE_ATTRS describes the attributes of the SQLJ object types accessible to the current user.

### Related Views

- DBA_SQLJ_TYPE_ATTRS describes the attributes of all SQLJ object types in the database.
- USER_SQLJ_TYPE_ATTRS describes the attributes of the object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the type |
| ATTR_NAME | VARCHAR2(128) | NOT NULL | Name of the attribute |
| EXTERNAL_ATTR_NAME | VARCHAR2(4000) | | External name of the attribute |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ATTR_TYPE_MOD | VARCHAR2(7) | | Type modifier of the attribute:<br>• REF<br>• POINTER |
| ATTR_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the attribute |
| ATTR_TYPE_NAME | VARCHAR2(128) | | Name of the type of the attribute |
| LENGTH | NUMBER | | Length of the CHAR attribute, or maximum length of the VARCHAR or VARCHAR2 attribute. |
| PRECISION | NUMBER | | Decimal precision of the NUMBER or DECIMAL attribute, or binary precision of the FLOAT attribute. |
| SCALE | NUMBER | | Scale of the NUMBER or DECIMAL attribute |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Character set name of the attribute (CHAR_CS or NCHAR_CS) |
| ATTR_NO | NUMBER | NOT NULL | Syntactical order number or position of the attribute as specified in the type specification or CREATE TYPE statement (not to be used as an ID number) |
| INHERITED | VARCHAR2(3) | | Indicates whether the attribute is inherited from a supertype (YES) or not (NO) |

**See Also:**

- "DBA_SQLJ_TYPE_ATTRS"
- "USER_SQLJ_TYPE_ATTRS"

# ALL_SQLJ_TYPE_METHODS

ALL_SQLJ_TYPE_METHODS describes the methods of the SQLJ object types accessible to the current user.

### Related Views

- DBA_SQLJ_TYPE_METHODS describes the methods of all SQLJ object types in the database.
- USER_SQLJ_TYPE_METHODS describes the methods of the SQLJ object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the type |
| METHOD_NAME | VARCHAR2(128) | NOT NULL | Name of the method |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXTERNAL_VAR_NAME | VARCHAR2(4000) | | Name of the external variable |
| METHOD_NO | NUMBER | NOT NULL | Method number that distinguishes overloaded methods (not to be used as an ID number) |
| METHOD_TYPE | VARCHAR2(6) | | Type of the method:<br>• MAP<br>• ORDER<br>• PUBLIC |
| PARAMETERS | NUMBER | NOT NULL | Number of parameters to the method |
| RESULTS | NUMBER | NOT NULL | Number of results returned by the method |
| FINAL | VARCHAR2(3) | | Indicates whether the method is final (YES) or not (NO) |
| INSTANTIABLE | VARCHAR2(3) | | Indicates whether the method is instantiable (YES) or not (NO) |
| OVERRIDING | VARCHAR2(3) | | Indicates whether the method is overriding a supertype method (YES) or not (NO) |
| INHERITED | VARCHAR2(3) | | Indicates whether the method is inherited from a supertype (YES) or not (NO) |

**See Also:**

- "DBA_SQLJ_TYPE_METHODS"
- "USER_SQLJ_TYPE_METHODS"

# ALL_SQLJ_TYPES

ALL_SQLJ_TYPES describes the SQLJ object types accessible to the current user.

### Related Views

- DBA_SQLJ_TYPES describes all SQLJ object types in the database.
- USER_SQLJ_TYPES describes the SQLJ object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the type |
| TYPE_OID | RAW(16) | NOT NULL | Object identifier (OID) of the type |
| EXTERNAL_NAME | VARCHAR2(4000) | | External class name of the type |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USING | VARCHAR2(21) | | Representation of the type:<br>• SQLData<br>• CustomDatum<br>• Serializable<br>• Serializable Internal<br>• ORAData |
| TYPECODE | VARCHAR2(128) | | Typecode of the type |
| ATTRIBUTES | NUMBER | | Number of attributes (if any) in the type |
| METHODS | NUMBER | | Number of methods (if any) in the type |
| PREDEFINED | VARCHAR2(3) | | Indicates whether the type is a predefined type (YES) or not (NO) |
| INCOMPLETE | VARCHAR2(3) | | Indicates whether the type is an incomplete type (YES) or not (NO) |
| FINAL | VARCHAR2(3) | | Indicates whether the type is a final type (YES) or not (NO) |
| INSTANTIABLE | VARCHAR2(3) | | Indicates whether the type is an instantiable type (YES) or not (NO) |
| SUPERTYPE_OWNER | VARCHAR2(128) | | Owner of the supertype (NULL if type is not a subtype) |
| SUPERTYPE_NAME | VARCHAR2(128) | | Name of the supertype (NULL if type is not a subtype) |
| LOCAL_ATTRIBUTES | NUMBER | | Number of local (not inherited) attributes (if any) in the subtype |
| LOCAL_METHODS | NUMBER | | Number of local (not inherited) methods (if any) in the subtype |

**See Also:**

- "DBA_SQLJ_TYPES"
- "USER_SQLJ_TYPES"

# ALL_SQLSET

ALL_SQLSET displays information about all SQL tuning sets accessible to the current user.

### Related Views

- DBA_SQLSET displays information about all SQL tuning sets in the database.
- USER_SQLSET displays information about the SQL tuning sets owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | NOT NULL | Name of the SQL tuning set |
| ID | NUMBER | NOT NULL | SQL tuning set identifier |
| CON_DBID | NUMBER | NOT NULL | The database ID of the PDB |
| OWNER | VARCHAR2(128) | | Owner of the SQL tuning set |
| DESCRIPTION | VARCHAR2(256) | | Description of the SQL tuning set |
| CREATED | DATE | | Date the SQL tuning set was created |
| LAST_MODIFIED | DATE | | Date the SQL tuning set was last modified |
| STATEMENT_COUNT | NUMBER | | Number of statements in the SQL tuning set |

**See Also:**

- "DBA_SQLSET"
- "USER_SQLSET"

# ALL_SQLSET_BINDS

ALL_SQLSET_BINDS displays the bind values associated with all SQL tuning sets accessible to the current user.

### Related Views

- DBA_SQLSET_BINDS displays the bind values associated with all SQL tuning sets in the database.
- USER_SQLSET_BINDS displays the bind values associated with the SQL tuning sets owned by the current user. This view does not display the SQLSET_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SQLSET_NAME | VARCHAR2(128) | | Name of the SQL tuning set for the statement |
| SQLSET_OWNER | VARCHAR2(128) | | User name of the SQL tuning set owner |
| SQLSET_ID | NUMBER | | ID of the SQL tuning set for the statement |
| CON_DBID | NUMBER | | The database ID of the PDB |
| SQL_ID | VARCHAR2(13) | | SQL identifier of the parent cursor in the library cache |
| FORCE_MATCHING_SIGNATURE | NUMBER | | The signature used when the CURSOR_SHARING parameter is set to FORCE |
| PLAN_HASH_VALUE | NUMBER | | Numerical representation of the SQL plan for the cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | are the same (rather than comparing the two plans line-by-line). |
| POSITION | NUMBER | | Bind position |
| VALUE | ANYDATA | | Bind value. This column is NULL for PL/SQL bind types. |
| CAPTURED | CHAR(1) | | Binds captured |
| SQL_SEQ | NUMBER | | SQL sequence |

**See Also:**

- "DBA_SQLSET_BINDS"
- "USER_SQLSET_BINDS"

# ALL_SQLSET_PLANS

ALL_SQLSET_PLANS describes captured plans for statements in the SQL tuning sets accessible to the current user.

### Related Views

- DBA_SQLSET_PLANS describes captured plans in the SQL tuning sets in the database.
- USER_SQLSET_PLANS describes captured plans for statements in the SQL tuning sets owned by the current user. This view does not display the SQLSET_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SQLSET_NAME | VARCHAR2(128) | NOT NULL | Name of SQL tuning set for the statement |
| SQLSET_OWNER | VARCHAR2(128) | | User name of SQL tuning set owner |
| SQLSET_ID | NUMBER | NOT NULL | ID of SQL tuning set for the statement |
| CON_DBID | NUMBER | NOT NULL | The database ID of the PDB |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| FORCE_MATCHING_SIGNATURE | NUMBER | NOT NULL | The signature used when the CURSOR_SHARING parameter is set to FORCE |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Numerical representation of the SQL plan for the cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line-by-line). |
| SQL_SEQ | NUMBER | NOT NULL | SQL sequence |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATEMENT_ID | VARCHAR2(30) | | Statement ID |
| PLAN_ID | NUMBER | | Plan ID |
| TIMESTAMP | DATE | | Date and time timestamp |
| REMARKS | VARCHAR2(4000) | | Remarks |
| OPERATION | VARCHAR2(30) | | Name of the internal operation performed in this step (for example, TABLE ACCESS) |
| OPTIONS | VARCHAR2(255) | | A variation on the operation described in the OPERATION column (for example, FULL) |
| OBJECT_NODE | VARCHAR2(128) | | Name of the database link used to reference the object (a table name or view name). For local queries that use parallel execution, this column describes the order in which output from operations is consumed. |
| OBJECT_OWNER | VARCHAR2(128) | | Name of the user who owns the schema containing the table or index |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or index |
| OBJECT_ALIAS | VARCHAR2(261) | | Alias for the object |
| OBJECT_INSTANCE | NUMBER(38) | | Instance number for the object |
| OBJECT_TYPE | VARCHAR2(30) | | Type of the object |
| OPTIMIZER | VARCHAR2(255) | | Current mode of the optimizer for the first row in the plan (statement line), for example, CHOOSE. When the operation is a database access (for example, TABLE ACCESS), this column indicates whether or not the object is analyzed. |
| SEARCH_COLUMNS | NUMBER | | Number of index columns with start and stop keys (that is, the number of columns with matching predicates) |
| ID | NUMBER(38) | NOT NULL | A number assigned to each step in the execution plan |
| PARENT_ID | NUMBER(38) | | ID of the next execution step that operates on the output of the current step |
| DEPTH | NUMBER(38) | | Depth (or level) of the operation in the tree. It is not necessary to issue a CONNECT BY statement to get the level information, which is generally used to indent the rows from the PLAN_TABLE table. The root operation (statement) is level 0. |
| POSITION | NUMBER(38) | | Order of processing for all operations that have the same PARENT_ID. |
| COST | NUMBER(38) | | Cost of the operation as estimated by the optimizer's cost-based approach. For statements |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | that use the rule-based approach, this column is NULL. |
| CARDINALITY | NUMBER(38) | | Estimate, made by the cost-based optimizer, of the number of rows produced by the operation |
| BYTES | NUMBER(38) | | Estimate, made by the cost-based optimizer, of the number of bytes produced by the operation |
| OTHER_TAG | VARCHAR2(255) | | Describes the contents of the OTHER column. For information about values, see *Oracle Database SQL Tuning Guide*. |
| PARTITION_START | VARCHAR2(255) | | Start partition of a range of accessed partitions |
| PARTITION_STOP | VARCHAR2(255) | | Stop partition of a range of accessed partitions |
| PARTITION_ID | NUMBER(38) | | Step that computes the pair of values of the PARTITION_START and PARTITION_STOP columns |
| OTHER | LONG | | Other information specific to the execution step that users may find useful. For information about values, see *Oracle Database SQL Tuning Guide*. |
| DISTRIBUTION | VARCHAR2(30) | | Stores the method used to distribute rows from producer query servers to consumer query servers |
| CPU_COST | NUMBER(38) | | CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| IO_COST | NUMBER(38) | | I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| TEMP_SPACE | NUMBER(38) | | Temporary space usage of the operation (sort or hash join) as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| ACCESS_PREDICATES | VARCHAR2(4000) | | Predicates used to locate rows in an access structure. For example, start or stop predicates for an index range scan. |
| FILTER_PREDICATES | VARCHAR2(4000) | | Predicates used to filter rows before producing them |
| PROJECTION | VARCHAR2(4000) | | Expressions produced by the operation |
| TIME | NUMBER(38) | | Elapsed time (in seconds) of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| QBLOCK_NAME | VARCHAR2(128) | | Name of the query block |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OTHER_XML | CLOB | | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML since multiple pieces of information can be stored there. This includes:<br>• Name of the schema against which the query was parsed<br>• Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan<br>• Indication of whether or not dynamic statistics were used to produce the plan<br>• The outline data, a set of optimizer hints that can be used to regenerate the same plan<br>For further information about values, see *Oracle Database SQL Tuning Guide*. |
| EXECUTIONS | NUMBER | | Number of times the plan has been executed |
| STARTS | NUMBER | | Number of times this operation has been started, accumulated over the past executions |
| OUTPUT_ROWS | NUMBER | | Number of rows produced by the row source, accumulated over the past executions |
| CR_BUFFER_GETS | NUMBER | | Number of buffers received in consistent mode, accumulated over the past executions. Buffers are usually retrieved in consistent mode for queries. |
| CU_BUFFER_GETS | NUMBER | | Number of buffers retrieved in current mode, accumulated over the past executions. Buffers are retrieved in current mode for statements such as INSERT, UPDATE, and DELETE. |
| DISK_READS | NUMBER | | Number of physical disk reads performed by the operation, accumulated over the past executions |
| DISK_WRITES | NUMBER | | Number of physical disk writes performed by the operation, accumulated over the past executions |
| ELAPSED_TIME | NUMBER | | Elapsed time (in microseconds) corresponding to this operation, accumulated over the past executions |
| LAST_STARTS | NUMBER | | Number of times this operation has been started, during the last execution |
| LAST_OUTPUT_ROWS | NUMBER | | Number of rows produced by the row source, during the last execution |
| LAST_CR_BUFFER_GETS | NUMBER | | Number of buffers retrieved in consistent mode, during the last execution. Buffers are usually retrieved in consistent mode for queries. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_CU_BUFFER_GETS | NUMBER | | Number of buffers retrieved in current mode, during the last execution. Buffers are retrieved in current mode for statements such as INSERT, UPDATE, and DELETE. |
| LAST_DISK_READS | NUMBER | | Number of physical disk reads performed by the operation, during the last execution |
| LAST_DISK_WRITES | NUMBER | | Number of physical disk writes performed by the operation, during the last execution |
| LAST_ELAPSED_TIME | NUMBER | | Elapsed time (in microseconds) corresponding to this operation, during the last execution |
| POLICY | VARCHAR2(10) | | Sizing policy for this work area:<br>• MANUAL<br>• AUTO |
| ESTIMATED_OPTIMAL_SIZE | NUMBER | | Estimated size (in KB) required by this work area to execute the operation completely in memory (optimal execution). This is either derived from optimizer statistics or from previous executions. |
| ESTIMATED_ONEPASS_SIZE | NUMBER | | Estimated size (in KB) required by this work area to execute the operation in a single pass. This is either derived from optimizer statistics or from previous executions. |
| LAST_MEMORY_USED | NUMBER | | Memory size (in KB) used by this work area during the last execution of the cursor |
| LAST_EXECUTION | VARCHAR2(10) | | Indicates whether this work area ran using OPTIMAL, ONE PASS, or under ONE PASS memory requirement (MULTI-PASS), during the last execution of the cursor |
| LAST_DEGREE | NUMBER | | Degree of parallelism used, during the last execution of the cursor |
| TOTAL_EXECUTIONS | NUMBER | | Number of times this work area was active |
| OPTIMAL_EXECUTIONS | NUMBER | | Number of times this work area ran in optimal mode |
| ONEPASS_EXECUTIONS | NUMBER | | Number of times this work area ran in one pass mode |
| MULTIPASSES_EXECUTIONS | NUMBER | | Number of times this work area ran below the one pass memory requirement |
| ACTIVE_TIME | NUMBER | | Average time this work area is active (in hundredths of a second) |
| MAX_TEMPSEG_SIZE | NUMBER | | Maximum temporary segment size (in bytes) created by an instantiation of this work area. This column is null if this work area has never spilled to disk. |
| LAST_TEMPSEG_SIZE | NUMBER | | Temporary segment size (in bytes) created in the last instantiation of this work area. This column |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | is null if the last instantiation of this work area did not spill to disk. |

**See Also:**

- "DBA_SQLSET_PLANS"
- "USER_SQLSET_PLANS"

# ALL_SQLSET_REFERENCES

ALL_SQLSET_REFERENCES describes whether or not the SQL tuning sets accessible to the current user are active.

### Related Views

- DBA_SQLSET_REFERENCES describes whether or not all SQL tuning sets in the database are active. A SQL tuning set cannot be dropped if it is referenced.
- USER_SQLSET_REFERENCES describes whether or not the SQL tuning sets owned by the current user are active.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SQLSET_NAME | VARCHAR2(128) | NOT NULL | Name of the SQL tuning set |
| SQLSET_OWNER | VARCHAR2(128) | | User name of SQL tuning set owner |
| SQLSET_ID | NUMBER | NOT NULL | Identifier of the SQL tuning set |
| ID | NUMBER | NOT NULL | Reference identifier |
| OWNER | VARCHAR2(128) | | User who registered to use the SQL tuning set |
| DESCRIPTION | VARCHAR2(256) | | Description of the usage of the SQL tuning set |
| CREATED | DATE | | Date the reference was created |

**See Also:**

- "DBA_SQLSET_REFERENCES"
- "USER_SQLSET_REFERENCES"

# ALL_SQLSET_STATEMENTS

ALL_SQLSET_STATEMENTS displays information about the SQL statements, along with their statistics, that form all SQL tuning sets accessible to the current user.

### Related Views

- DBA_SQLSET_STATEMENTS displays information about the SQL statements, along with their statistics, that form all SQL tuning sets in the database.
- USER_SQLSET_STATEMENTS displays information about the SQL statements, along with their statistics, that form the SQL tuning sets owned by the current user. This view does not display the SQLSET_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SQLSET_NAME | VARCHAR2(128) | NOT NULL | Name of the SQL tuning set for the statement |
| SQLSET_OWNER | VARCHAR2(128) | | User name of the SQL tuning set owner |
| SQLSET_ID | NUMBER | NOT NULL | ID of the SQL tuning set for the statement |
| CON_DBID | NUMBER | NOT NULL | The database ID of the PDB |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| FORCE_MATCHING_SIGNATURE | NUMBER | NOT NULL | The signature used when the CURSOR_SHARING parameter is set to FORCE |
| SQL_TEXT | CLOB | | Full text for the SQL statement exposed as a CLOB column. |
| PARSING_SCHEMA_NAME | VARCHAR2(128) | | Name of the user in whose schema the statement was parsed |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Hash value for the plan corresponding to statistics in this record |
| BIND_DATA | RAW(2000) | | Bind data |
| BINDS_CAPTURED | CHAR(1) | | Binds captured |
| MODULE | VARCHAR2(64) | | Contains the name of the module that was executing when the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_MODULE |
| ACTION | VARCHAR2(64) | | Contains the name of the action that was executing when the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_ACTION |
| ELAPSED_TIME | NUMBER | | Elapsed time (in microseconds) used by this cursor for parsing, executing, and fetching |
| CPU_TIME | NUMBER | | CPU time (in microseconds) used by this cursor for parsing, executing, and fetching |
| BUFFER_GETS | NUMBER | | Number of buffer gets for this child cursor |
| DISK_READS | NUMBER | | Number of disk reads for this child cursor |
| DIRECT_WRITES | NUMBER | | Number of direct writes for this child cursor |
| ROWS_PROCESSED | NUMBER | | Total number of rows that the parsed SQL statement returns |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FETCHES | NUMBER | | Number of fetches associated with the SQL statement |
| EXECUTIONS | NUMBER | | Number of executions that took place on this object since it was brought into the library cache |
| END_OF_FETCH_COUNT | NUMBER | | Number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic in not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less than, or equal to, the value of the EXECUTIONS column. |
| OPTIMIZER_COST | NUMBER | | Cost of this query, given by the optimizer |
| OPTIMIZER_ENV | RAW(2000) | | Optimizer environment |
| PRIORITY | NUMBER | | User-defined priority |
| COMMAND_TYPE | NUMBER | | Oracle command type definition |
| FIRST_LOAD_TIME | VARCHAR2(19) | | Timestamp of the parent creation time |
| STAT_PERIOD | NUMBER | | Time (in seconds) during which the statistics of the SQL statement were collected |
| ACTIVE_STAT_PERIOD | NUMBER | | Effective time (in seconds) during which the SQL statement was active |
| OTHER | CLOB | | Client data, specified by the user, for this statement |
| PLAN_TIMESTAMP | DATE | | Timestamp for the plan corresponding to the statistics in this record |
| SQL_SEQ | NUMBER | NOT NULL | SQL sequence |

**See Also:**

- "DBA_SQLSET_STATEMENTS"
- "USER_SQLSET_STATEMENTS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_MODULE procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_ACTION procedure

# ALL_STAT_EXTENSIONS

ALL_STAT_EXTENSIONS displays information about the optimizer statistics extensions accessible to the current user.

### Related Views

- DBA_STAT_EXTENSIONS displays information about all optimizer statistics extensions in the database.
- USER_STAT_EXTENSIONS displays information about the optimizer statistics extensions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the extension |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table to which the extension belongs |
| EXTENSION_NAME | VARCHAR2(128) | NOT NULL | Name of the statistics extension |
| EXTENSION | CLOB | | Extension (the expression or column group) |
| CREATOR | VARCHAR2(6) | | Creator of the extension:<br>• USER<br>• SYSTEM |
| DROPPABLE | VARCHAR2(3) | | Indicates whether the extension is droppable using DBMS_STATS.DROP_EXTENDED_STATS (YES) or not (NO) |

**See Also:**

- "DBA_STAT_EXTENSIONS"
- "USER_STAT_EXTENSIONS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS.DROP_EXTENDED_STATS procedure

# ALL_STORED_SETTINGS

ALL_STORED_SETTINGS provides information about the persistent parameter settings for stored PL/SQL units for which the current user has execute privileges.

### Related Views

- DBA_STORED_SETTINGS lists information about the persistent parameter settings for stored PL/SQL units for which the current user has execute privileges. It also returns parameter information for all objects in the database and is accessible only to users with the SELECT_CATALOG_ROLE privilege.

- USER_STORED_SETTINGS lists information about the persistent parameter settings for stored PL/SQL units, but only shows information about PL/SQL units owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Name of the database user owning the stored PL/SQL unit |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the PL/SQL unit |
| OBJECT_ID | NUMBER | NOT NULL | Object number of the PL/SQL unit |
| OBJECT_TYPE | VARCHAR2(12) | | The type of PL/SQL unit: PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, TRIGGER, TYPE, or TYPE BODY |
| PARAM_NAME | VARCHAR2(128) | NOT NULL | The name of the parameter stored persistently with the PL/SQL unit |
| PARAM_VALUE | VARCHAR2(4000) | | The TO_CHAR() representation of the value of the persistently stored parameter. The width of this column is operating system dependent; however, it is never less than 255. |
| ORIGIN_CON_ID[1] | NUMBER | | ID of the container where the row originates |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

This view is deprecated in favor of the ALL_PLSQL_OBJECT_SETTINGS view. Oracle recommends that you use ALL_PLSQL_OBJECT_SETTINGS instead. ALL_STORED_SETTINGS is retained for backward compatibility only.

**See Also:**

"ALL_PLSQL_OBJECT_SETTINGS"

# ALL_STREAMS_COLUMNS

ALL_STREAMS_COLUMNS displays information about the Streams unsupported columns accessible to the current user.

### Related View

DBA_STREAMS_COLUMNS displays information about all Streams unsupported columns in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SYNC_CAPTURE_VERSION | NUMBER | | Version of the synchronous capture that supports this column |
| SYNC_CAPTURE_REASON | VARCHAR2(39) | | Reason why this column is not supported by synchronous capture |
| APPLY_VERSION | NUMBER | | Version of apply that supports this column |
| APPLY_REASON | VARCHAR2(40) | | Reason why this column is not supported by apply |

**See Also:**

"DBA_STREAMS_COLUMNS"

# ALL_STREAMS_GLOBAL_RULES

ALL_STREAMS_GLOBAL_RULES displays information about the following types of rules:

- Global rules created for the Streams capture processes that enqueue the captured changes into queues accessible to the current user
- Global rules created for the Streams propagations that have a source queue accessible to the current user
- Global rules created for the Streams apply processes that dequeue events from queues accessible to the current user

This view only contains information about rules created using the ADD_GLOBAL_RULES or ADD_GLOBAL_PROPAGATION_RULES procedures in the DBMS_STREAMS_ADM package. It does not contain information about rules created using the DBMS_RULE_ADM package.

### Related View

DBA_STREAMS_GLOBAL_RULES displays information about the global rules created for all Streams capture processes, propagations, and apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAMS_NAME | VARCHAR2(128) | | Name of the Streams process or propagation |
| STREAMS_TYPE | VARCHAR2(11) | | Type of the Streams process or propagation:<br>• CAPTURE<br>• PROPAGATION<br>• APPLY |
| RULE_TYPE | VARCHAR2(9) | | Type of the rule:<br>• DML<br>• DDL |
| INCLUDE_TAGGED_LCR | VARCHAR2(3) | | Indicates whether a redo entry or logical change record (LCR) with a non-NULL tag is considered |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | for capture, propagation, or apply (YES) or not (NO) |
| SOURCE_DATABASE | VARCHAR2(128) | | Source database in the rule condition. The rule evaluates to true for a redo entry or logical change record (LCR) only if the redo entry or LCR contains this source database. |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_CONDITION | VARCHAR2(4000) | | First 4000 bytes of the system-generated rule condition evaluated by the rules engine |

**See Also:**

- "DBA_STREAMS_GLOBAL_RULES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STREAMS_ADM package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_RULE_ADM package

# ALL_STREAMS_MESSAGE_CONSUMERS

ALL_STREAMS_MESSAGE_CONSUMERS displays information about the Streams messaging clients accessible to the current user.

### Related View

DBA_STREAMS_MESSAGE_CONSUMERS displays information about all Streams messaging clients in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAMS_NAME | VARCHAR2(128) | NOT NULL | Name of the messaging client |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| QUEUE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the positive rule set |
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the positive rule set |
| NEGATIVE_RULE_SET_NAME | VARCHAR2(128) | | Name of the negative rule set |
| NEGATIVE_RULE_SET_OWNER | VARCHAR2(128) | | Owner of the negative rule set |
| NOTIFICATION_TYPE | VARCHAR2(9) | | Type of the notification action:<br>• PROCEDURE<br>• MAIL |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • HTTP |
| NOTIFICATION_ACTION | VARCHAR2(256) | | Notification action |
| NOTIFICATION_CONTEXT | ANYDATA | | Context for the notification action |

**See Also:**

"DBA_STREAMS_MESSAGE_CONSUMERS"

# ALL_STREAMS_MESSAGE_RULES

ALL_STREAMS_MESSAGE_RULES displays information about the Streams messaging rules accessible to the current user.

### Related View

DBA_STREAMS_MESSAGE_RULES displays information about all Streams messaging rules in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAMS_NAME | VARCHAR2(128) | | Name of the Streams client |
| STREAMS_TYPE | VARCHAR2(11) | | Type of the Streams client:<br>• PROPAGATION - Streams propagation<br>• APPLY - Streams apply process<br>• DEQUEUE - Streams messaging client |
| MESSAGE_TYPE_NAME | VARCHAR2(128) | | Name of the message type |
| MESSAGE_TYPE_OWNER | VARCHAR2(128) | | Owner of the message type |
| MESSAGE_RULE_VARIABLE | VARCHAR2(128) | | Name of the variable in the message rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_CONDITION | VARCHAR2(4000) | | Rule condition |

**See Also:**

- "DBA_STREAMS_MESSAGE_RULES"

# ALL_STREAMS_NEWLY_SUPPORTED

ALL_STREAMS_NEWLY_SUPPORTED displays information about the tables accessible to the current user that are newly supported by Streams capture processes.

### Related View

DBA_STREAMS_NEWLY_SUPPORTED displays information about all tables in the database that are newly supported by Streams capture processes.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| REASON | VARCHAR2(39) | | Reason why the table was not supported in a previous release:<br>• IOT<br>• column with user-defined type<br>• unsupported column exists<br>• object table<br>• AQ queue table<br>• temporary table<br>• sub object<br>• external table<br>• materialized view<br>• FILE column exists<br>• materialized view log<br>• materialized view container table<br>• streams unsupported object<br>• domain index |
| COMPATIBLE | CHAR(4) | | Minimum database compatibility for capture processes to support the database object |

**See Also:**

"DBA_STREAMS_NEWLY_SUPPORTED"

# ALL_STREAMS_RULES

ALL_STREAMS_RULES displays information about the rules used by the Streams processes accessible to the current user.

### Related View

DBA_STREAMS_RULES displays information about the rules used by all Streams processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAMS_TYPE | VARCHAR2(12) | | Type of the Streams process:<br>• CAPTURE<br>• PROPAGATION<br>• APPLY<br>• DEQUEUE<br>• SYNC_CAPTURE |
| STREAMS_NAME | VARCHAR2(128) | | Name of the Streams process |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the rule set |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the rule set |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_CONDITION | CLOB | | Current rule condition |
| RULE_SET_TYPE | CHAR(8) | | Type of the rule set:<br>• POSITIVE<br>• NEGATIVE |
| STREAMS_RULE_TYPE | VARCHAR2(9) | | For global, schema, or table rules, the type of the rule:<br>• TABLE<br>• SCHEMA<br>• GLOBAL |
| SCHEMA_NAME | VARCHAR2(128) | | For table and schema rules, the schema name |
| OBJECT_NAME | VARCHAR2(128) | | For table rules, the table name |
| SUBSETTING_OPERATION | VARCHAR2(6) | | For subset rules, the type of operation:<br>• INSERT<br>• UPDATE<br>• DELETE |
| DML_CONDITION | VARCHAR2(4000) | | For subset rules, the row subsetting condition |
| INCLUDE_TAGGED_LCR | VARCHAR2(3) | | For global, schema, or table rules, indicates whether to include tagged logical change records (LCRs) (YES) or not (NO) |
| SOURCE_DATABASE | VARCHAR2(128) | | For global, schema, or table rules, the global name of the database where the logical change records (LCRs) originated |
| RULE_TYPE | VARCHAR2(9) | | For global, schema, or table rules, the type of the rule:<br>• DML<br>• DDL |
| MESSAGE_TYPE_OWNER | VARCHAR2(128) | | For message rules, the owner of the message type |
| MESSAGE_TYPE_NAME | VARCHAR2(128) | | For message rules, the name of the message type |
| MESSAGE_RULE_VARIABLE | VARCHAR2(128) | | For message rules, the name of the variable in the message rule |
| ORIGINAL_RULE_CONDITION | VARCHAR2(4000) | | For rules created by Streams administrative APIs, the original rule condition when the rule was created |
| SAME_RULE_CONDITION | VARCHAR2(3) | | For rules created by Streams administrative APIs, indicates whether the current rule condition is |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | the same as the original rule condition (YES) or not (NO) |

**See Also:**

"DBA_STREAMS_RULES"

# ALL_STREAMS_SCHEMA_RULES

ALL_STREAMS_SCHEMA_RULES displays information about the following types of rules:

- Schema rules created for the Streams capture processes that enqueue the captured changes into queues accessible to the current user
- Schema rules created for the Streams propagations that have a source queue accessible to the current user
- Schema rules created for the Streams apply processes that dequeue events from queues accessible to the current user

This view only contains information about rules created using the ADD_SCHEMA_RULES or ADD_SCHEMA_PROPAGATION_RULES procedures in the DBMS_STREAMS_ADM package. It does not contain information about rules created using the DBMS_RULE_ADM package.

### Related View

DBA_STREAMS_SCHEMA_RULES displays information about the schema rules created for all Streams capture processes, propagations, and apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAMS_NAME | VARCHAR2(128) | | Name of the Streams process or propagation |
| STREAMS_TYPE | VARCHAR2(11) | | Type of the Streams process or propagation:<br>• CAPTURE<br>• PROPAGATION<br>• APPLY |
| SCHEMA_NAME | VARCHAR2(128) | | Schema name in the rule condition. The rule evaluates to true for a redo entry or logical change record (LCR) only if the redo entry or LCR contains this schema name. |
| RULE_TYPE | VARCHAR2(7) | | Type of the rule:<br>• DML<br>• DDL |
| INCLUDE_TAGGED_LCR | VARCHAR2(3) | | Indicates whether a redo entry or logical change record (LCR) with a non-NULL tag is considered for capture, propagation, or apply (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_DATABASE | VARCHAR2(128) | | Source database in the rule condition. The rule evaluates to `true` for a redo entry or logical change record (LCR) only if the redo entry or LCR contains this source database. |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_CONDITION | VARCHAR2(4000) | | First 4000 bytes of the system-generated rule condition evaluated by the rules engine |

**See Also:**

- "DBA_STREAMS_SCHEMA_RULES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STREAMS_ADM package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_RULE_ADM package

# ALL_STREAMS_TABLE_RULES

ALL_STREAMS_TABLE_RULES displays information about the following types of rules:

- Table rules created for the Streams capture processes that enqueue the captured changes into queues accessible to the current user
- Table rules created for the Streams propagations that have a source queue accessible to the current user
- Table rules created for the Streams apply processes that dequeue events from queues accessible to the current user
- Subset rules created for the Streams apply processes that have a source queue accessible to the current user

This view only contains information about rules created using the ADD_TABLE_RULES, ADD_TABLE_PROPAGATION_RULES, or ADD_SUBSET_RULES procedures in the DBMS_STREAMS_ADM package. It does not contain information about rules created using the DBMS_RULE_ADM package.

### Related View

DBA_STREAMS_TABLE_RULES displays information about the table rules created for all Streams capture processes, propagations, and apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAMS_NAME | VARCHAR2(128) | | Name of the Streams process or propagation |
| STREAMS_TYPE | VARCHAR2(12) | | Type of the Streams process or propagation:<br>• CAPTURE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • PROPAGATION<br>• APPLY<br>• DEQUEUE<br>• SYNC_CAPTURE |
| TABLE_OWNER | VARCHAR2(128) | | Table owner in the rule condition. The rule evaluates to `true` for a redo entry or logical change record (LCR) only if the redo entry or LCR contains this table owner. |
| TABLE_NAME | VARCHAR2(128) | | Table name in the rule condition. The rule evaluates to `true` for a redo entry or logical change record (LCR) only if the redo entry or LCR contains this table name. |
| RULE_TYPE | VARCHAR2(7) | | Type of the rule:<br>• DML<br>• DDL |
| DML_CONDITION | VARCHAR2(4000) | | Row subsetting condition, if the rule is a subset rule |
| SUBSETTING_OPERATION | VARCHAR2(6) | | DML operation for row subsetting in the rule condition, if the rule is a subset rule:<br>• INSERT<br>• UPDATE<br>• DELETE<br>The rule evaluates to `true` for a logical change record (LCR) only if the LCR contains this command type after internal transformation. |
| INCLUDE_TAGGED_LCR | VARCHAR2(3) | | Indicates whether a redo entry or logical change record (LCR) with a non-NULL tag is considered for capture, propagation, or apply (YES) or not (NO) |
| SOURCE_DATABASE | VARCHAR2(128) | | Source database in the rule condition. The rule evaluates to `true` for a redo entry or logical change record (LCR) only if the redo entry or LCR contains this source database. |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_CONDITION | VARCHAR2(4000) | | First 4000 bytes of the system-generated rule condition evaluated by the rules engine |

**See Also:**

- "DBA_STREAMS_TABLE_RULES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STREAMS_ADM package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_RULE_ADM package

# ALL_STREAMS_TRANSFORM_FUNCTION

ALL_STREAMS_TRANSFORM_FUNCTION displays information about the rule-based transformation functions accessible to the current user.

### Related View

DBA_STREAMS_TRANSFORM_FUNCTION displays information about all rule-based transformation functions in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule associated with the transformation function |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule associated with the transformation function |
| VALUE_TYPE | VARCHAR2(4000) | | Type of the transformation function name. This type must be VARCHAR2 for a rule-based transformation to work properly. |
| TRANSFORM_FUNCTION_NAME | VARCHAR2(4000) | | Name of the transformation function (NULL if VALUE_TYPE is not VARCHAR2) |
| CUSTOM_TYPE | VARCHAR2(11) | | Type of the transformation function:<br>• ONE TO ONE - One-to-one transformations<br>• ONE TO MANY - One-to-many transformations |

**See Also:**

- "DBA_STREAMS_TRANSFORM_FUNCTION"
- *Oracle Streams Concepts and Administration* for more information about custom rule-based transformation

# ALL_STREAMS_UNSUPPORTED

ALL_STREAMS_UNSUPPORTED displays information about the tables accessible to the current user that are not supported by Streams in this release of the Oracle Database.

### Related View

DBA_STREAMS_UNSUPPORTED displays information about all tables in the database that are not supported by Streams in this release of the Oracle Database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| REASON | VARCHAR2(39) | | Reason why the table is not supported:<br>• IOT<br>• column with user-defined type<br>• unsupported column exists<br>• object table<br>• AQ queue table<br>• temporary table<br>• sub object<br>• external table<br>• materialized view<br>• FILE column exists<br>• materialized view log<br>• materialized view container table<br>• streams unsupported object<br>• domain index<br>• IOT with overflow<br>• IOT with LOB<br>• IOT with physical Rowid mapping<br>• mapping table for physical rowid of IOT<br>• IOT with LOB<br>• IOT with row movement<br>• summary container table |
| AUTO_FILTERED | VARCHAR2(3) | | Indicates whether Streams automatically filters out the object (YES) or not (NO) |

**See Also:**

"DBA_STREAMS_UNSUPPORTED"

# ALL_SUBPART_COL_STATISTICS

ALL_SUBPART_COL_STATISTICS describes column statistics and histogram information for subpartitions of partitioned objects accessible to the current user.

### Related Views

- DBA_SUBPART_COL_STATISTICS provides this information for all subpartitions in the database.
- USER_SUBPART_COL_STATISTICS provides this information for subpartitions of all partitioned objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| SUBPARTITION_NAME | VARCHAR2(128) | | Table subpartition name |
| COLUMN_NAME | VARCHAR2(4000) | | Column name |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column |
| HIGH_VALUE | RAW(1000) | | High value in the column |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/NUM_DISTINCT. |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| NUM_BUCKETS | NUMBER | | Number of buckets in histogram for the column |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this column |
| LAST_ANALYZED | DATE | | Date on which this column was most recently analyzed |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered on a subpartition (YES) or they were not gathered (NO) |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| NOTES | VARCHAR2(41) | | Describes some additional properties of the statistics. For example, if the value is INCREMENTAL, the global statistics are derived from synopses, that is, the global statistics are incrementally maintained. |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE<br>• FREQUENCY<br>• HEIGHT BALANCED<br>• HYBRID<br>• TOP-FREQUENCY |

**See Also:**

- "DBA_SUBPART_COL_STATISTICS"
- "USER_SUBPART_COL_STATISTICS"

# ALL_SUBPART_HISTOGRAMS

ALL_SUBPART_HISTOGRAMS displays the actual histogram data (end-points per histogram) for histograms on table subpartitions accessible to the current user.

### Related Views

- DBA_SUBPART_HISTOGRAMS displays this information for all subpartitions in the database.
- USER_SUBPART_HISTOGRAMS displays this information for subpartitions of all partitioned objects owned by the current user. This view does not display the OWNER column.

**Note:**

These views are populated only if you collect statistics on the index using the DBMS_STATS package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| SUBPARTITION_NAME | VARCHAR2(128) | | Table subpartition name |
| COLUMN_NAME | VARCHAR2(4000) | | Column name |
| BUCKET_NUMBER | NUMBER | | Bucket number |
| ENDPOINT_VALUE | NUMBER | | Normalized endpoint values for this bucket |
| ENDPOINT_ACTUAL_VALUE | VARCHAR2(4000) | | Actual (not normalized) string value of the endpoint for this bucket |
| ENDPOINT_ACTUAL_VALUE_RAW | RAW(1000) | | Endpoint actual value in raw format |
| ENDPOINT_REPEAT_COUNT | NUMBER | | Frequency of the endpoint (applies only to hybrid histograms, and is set to 0 for other histogram types) |

**See Also:**

- "DBA_SUBPART_HISTOGRAMS"
- "USER_SUBPART_HISTOGRAMS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_SUBPART_KEY_COLUMNS

ALL_SUBPART_KEY_COLUMNS displays subpartitioning key columns for composite-partitioned tables (and local indexes on composite-partitioned tables) accessible to the current user.

### Related Views

- DBA_SUBPART_KEY_COLUMNS displays this information for all subpartitions in the database.
- USER_SUBPART_KEY_COLUMNS displays this information for subpartitions of all partitioned objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the partitioned table or index |
| NAME | VARCHAR2(128) | | Name of the partitioned table or index |
| OBJECT_TYPE | CHAR(5) | | Object type:<br>• TABLE<br>• INDEX |
| COLUMN_NAME | VARCHAR2(4000) | | Column name |
| COLUMN_POSITION | NUMBER | | Position of the column within the subpartitioning key |

**See Also:**

- "DBA_SUBPART_KEY_COLUMNS"
- "USER_SUBPART_KEY_COLUMNS"

# ALL_SUBPARTITION_TEMPLATES

ALL_SUBPARTITION_TEMPLATES describes the subpartition templates accessible to the current user.

### Related Views

- DBA_SUBPARTITION_TEMPLATES describes all subpartition templates in the database.
- USER_SUBPARTITION_TEMPLATES describes the subpartition templates owned by the current user. This view does not display the USER_NAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_NAME | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| SUBPARTITION_NAME | VARCHAR2(132) | NOT NULL | Name of the subpartition |
| SUBPARTITION_POSITION | NUMBER | | Position of the subpartition |
| TABLESPACE_NAME | VARCHAR2(30) | | Tablespace name of the subpartition |
| HIGH_BOUND | LONG | | Literal list values of the subpartition |

**See Also:**

- "DBA_SUBPARTITION_TEMPLATES"
- "USER_SUBPARTITION_TEMPLATES"

# ALL_SUMDELTA

ALL_SUMDELTA lists direct path load entries accessible to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLEOBJ# | NUMBER | NOT NULL | Object number of the table |
| PARTITIONOBJ# | NUMBER | NOT NULL | Object number of table partitions (if the table is partitioned) |
| DMLOPERATION | VARCHAR2(1) | | Type of DML operation applied to the table |
| SCN | NUMBER | NOT NULL | SCN when the bulk DML occurred |
| TIMESTAMP | DATE | NOT NULL | Timestamp of the log entry |
| LOWROWID | ROWID | NOT NULL | Start ROWID in the loaded rowid range |
| HIGHROWID | ROWID | NOT NULL | End ROWID in the loaded rowid range |
| SEQUENCE | NUMBER | | Sequence number of the direct load |
| XID | NUMBER | | Transaction ID |

# ALL_SYNC_CAPTURE

ALL_SYNC_CAPTURE displays information about the synchronous capture processes that store the captured changes in queues accessible to the current user.

### Related View

DBA_SYNC_CAPTURE displays information about all synchronous capture processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the capture process |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue used for holding captured changes |
| QUEUE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue used for holding captured changes |
| RULE_SET_NAME | VARCHAR2(128) | | Rule set used by the capture process |
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the rule set |
| CAPTURE_USER | VARCHAR2(128) | | Current user who is enqueuing captured messages |

**See Also:**

"DBA_SYNC_CAPTURE"

## ALL_SYNC_CAPTURE_PREPARED_TABS

ALL_SYNC_CAPTURE_PREPARED_TABS displays information about the tables accessible to the current user that are prepared for synchronous capture instantiation.

### Related View

DBA_SYNC_CAPTURE_PREPARED_TABS displays information about all tables in the database that are prepared for synchronous capture instantiation.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the table prepared for synchronous capture instantiation |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table prepared for synchronous capture instantiation |
| SCN | NUMBER | NOT NULL | SCN from which changes can be captured |
| TIMESTAMP | DATE | | Time at which the table was ready to be instantiated |

**See Also:**

"DBA_SYNC_CAPTURE_PREPARED_TABS"

# ALL_SYNC_CAPTURE_TABLES

ALL_SYNC_CAPTURE_TABLES displays information about the tables accessible to the current user that are captured by synchronous Streams captures.

### Related View

DBA_SYNC_CAPTURE_TABLES displays information about all tables in the database that are captured by synchronous Streams captures.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | | Owner of the synchronous capture table |
| TABLE_NAME | VARCHAR2(128) | | Name of the synchronous capture table |
| ENABLED | VARCHAR2(3) | | Indicates whether synchronous Streams capture is enabled for the table (YES) or not (NO) |

**See Also:**

"DBA_SYNC_CAPTURE_TABLES"

# ALL_SYNONYMS

ALL_SYNONYMS describes the synonyms accessible to the current user. The following criteria determine the list of synonyms that ALL_SYNONYMS shows:

- All private synonyms owned by the logged-in user, even if the base object pointed to is not accessible.
- All public synonyms, even if the base object pointed to is not accessible.
- All private synonyms owned by a different user, where the ultimate base object pointed to by that synonym or by any chain of nested synonyms, is know to be accessible because of a grant to the logged-in user, or a grant to a role in effect for this session.
- If the current session has any of the following privileges, then all synonyms that point directly to local objects are shown because it is assumed that the session can access those objects:
  - LOCK ANY TABLE
  - READ ANY TABLE
  - SELECT ANY TABLE
  - INSERT ANY TABLE
  - UPDATE ANY TABLE
  - DELETE ANY TABLE

  Synonyms that point to remote objects are excluded because the system privileges just listed do not automatically convey access to those remote objects. Also, if the

synonyms point to objects other than tables and views (such as sequences, PL/SQL procedures, and so on) then this rule may show synonyms that ultimately resolve to objects that this session cannot access.

- All private synonyms owned by a different user, where the synonym is via a database link, are excluded.

### Related Views

- DBA_SYNONYMS describes all synonyms in the database.
- USER_SYNONYMS describes the synonyms owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the synonym |
| SYNONYM_NAME | VARCHAR2(128) | | Name of the synonym |
| TABLE_OWNER | VARCHAR2(128) | | Owner of the object referenced by the synonym, or creator of the referring synonym if the target is a public synonym (that is, the object referred to by TABLE_NAME).<br>Although the column is called TABLE_OWNER, the object owned is not necessarily a table. It can be any general object such as a view, sequence, stored procedure, synonym, and so on. |
| TABLE_NAME | VARCHAR2(128) | | Name of the object referenced by the synonym. Although the column is called TABLE_NAME, the object does not necessarily have to be a table. It can be any general object such as a view, sequence, stored procedure, synonym, and so on. |
| DB_LINK | VARCHAR2(128) | | Name of the database link referenced, if any |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_SYNONYMS"
- "USER_SYNONYMS"

# ALL_TAB_COL_STATISTICS

ALL_TAB_COL_STATISTICS displays column statistics and histogram information extracted from "ALL_TAB_COLUMNS".

### Related Views

- DBA_TAB_COL_STATISTICS displays such information extracted from "DBA_TAB_COLUMNS".
- USER_TAB_COL_STATISTICS displays such information extracted from "USER_TAB_COLUMNS". This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| COLUMN_NAME | VARCHAR2(128) | | Column name |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column |
| HIGH_VALUE | RAW(1000) | | High value in the column |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/NUM_DISTINCT. |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| NUM_BUCKETS | NUMBER | | Number of buckets in histogram for the column |
| LAST_ANALYZED | DATE | | Date on which this column was most recently analyzed |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this column |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| NOTES | VARCHAR2(41) | | Describes some additional properties of the statistics. For example, if the value is INCREMENTAL, the global statistics are derived from synopses, that is, the global statistics are incrementally maintained. |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • FREQUENCY<br>• HEIGHT BALANCED<br>• HYBRID<br>• TOP-FREQUENCY |
| SCOPE | VARCHAR2(7) | | The value is SHARED for statistics gathered on any table other than global temporary tables.<br>For a global tempoary table, the possible values are:<br>• SESSION - Indicates that the statistics are session-specific<br>• SHARED - Indicates that the statistics are shared across all sessions<br>See *Oracle Database PL/SQL Packages and Types Reference* for information about using the GLOBAL_TEMP_TABLE_STATS preference of the DBMS_STATS package to control whether to gather session or shared statistics for global temporary tables. |

**See Also:**

- "DBA_TAB_COL_STATISTICS"
- "USER_TAB_COL_STATISTICS"

# ALL_TAB_COLS

ALL_TAB_COLS describes the columns of the tables, views, and clusters accessible to the current user. To gather statistics for this view, use the DBMS_STATS package.

This view differs from "ALL_TAB_COLUMNS" in that system-generated hidden columns and invisible columns, which are user-generated hidden columns, are not filtered out.

**Note:**

See *Oracle Database Administrator's Guide* for more information about invisible columns

### Related Views

- DBA_TAB_COLS describes the columns of all tables, views, and clusters in the database.
- USER_TAB_COLS describes the columns of the tables, views, and clusters owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table, view, or cluster |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table, view, or cluster |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name |
| DATA_TYPE | VARCHAR2(128) | | Data type of the column |
| DATA_TYPE_MOD | VARCHAR2(3) | | Data type modifier of the column |
| DATA_TYPE_OWNER | VARCHAR2(128) | | Owner of the data type of the column |
| DATA_LENGTH | NUMBER | NOT NULL | Length of the column (in bytes) |
| DATA_PRECISION | NUMBER | | Decimal precision for NUMBER datatype; binary precision for FLOAT datatype; NULL for all other datatypes |
| DATA_SCALE | NUMBER | | Digits to the right of the decimal point in a number |
| NULLABLE | VARCHAR2(1) | | Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. |
| COLUMN_ID | NUMBER | | Sequence number of the column as created |
| DEFAULT_LENGTH | NUMBER | | Length of the default value for the column |
| DATA_DEFAULT | LONG | | Default value for the column |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column[1] |
| HIGH_VALUE | RAW(1000) | | High value in the column[1] |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/NUM_DISTINCT.[1] |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| NUM_BUCKETS | NUMBER | | Number of buckets in the histogram for the column<br>**Note:** The number of buckets in a histogram is specified in the SIZE parameter of the ANALYZE SQL statement. However, Oracle Database does not create a histogram with more buckets than the number of rows in the sample. Also, if the sample contains any values that are very repetitious, Oracle Database creates the specified number of buckets, but the value indicated by this column may be smaller because of an internal compression algorithm. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_ANALYZED | DATE | | Date on which this column was most recently analyzed |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this column |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Name of the character set:<br>• CHAR_CS<br>• NCHAR_CS |
| CHAR_COL_DECL_LENGTH | NUMBER | | Declaration length of the character type column |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| CHAR_LENGTH | NUMBER | | Displays the length of the column in characters. This value only applies to the following datatypes:<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| CHAR_USED | VARCHAR2(1) | | Indicates that the column uses BYTE length semantics (B) or CHAR length semantics (C), or whether the datatype is not any of the following (NULL):<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| V80_FMT_IMAGE | VARCHAR2(3) | | Indicates whether the column data is in release 8.0 image format (YES) or not (NO) |
| DATA_UPGRADED | VARCHAR2(3) | | Indicates whether the column data has been upgraded to the latest type version format (YES) or not (NO) |
| HIDDEN_COLUMN | VARCHAR2(3) | | Indicates whether the column is a hidden column (YES) or not (NO) |
| VIRTUAL_COLUMN | VARCHAR2(3) | | Indicates whether the column is a virtual column (YES) or not (NO) |
| SEGMENT_COLUMN_ID | NUMBER | | Sequence number of the column in the segment |
| INTERNAL_COLUMN_ID | NUMBER | NOT NULL | Internal sequence number of the column |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE<br>• FREQUENCY<br>• HEIGHT BALANCED |
| QUALIFIED_COL_NAME | VARCHAR2(4000) | | Qualified column name |
| USER_GENERATED | VARCHAR2(3) | | Indicates whether the column is a user-generated column (YES) or a system-generated column (NO).<br>Invisible columns are hidden columns that are also user- generated columns. |
| DEFAULT_ON_NULL | VARCHAR2(3) | | Indicates whether the column has DEFAULT ON NULL semantics (YES) or not (NO) |
| IDENTITY_COLUMN | VARCHAR2(3) | | Indicates whether this is an identity column (YES) or not (NO) |
| EVALUATION_EDITION | VARCHAR2(128) | | Name of the edition in which editioned objects referenced in an expression column are resolved |
| UNUSABLE_BEFORE | VARCHAR2(128) | | Name of the oldest edition in which the column is usable |
| UNUSABLE_BEGINNING | VARCHAR2(128) | | Name of the oldest edition in which the column becomes perpetually unusable |

[1] These columns remain for backward compatibility with Oracle7. This information is now in the {TAB|PART}_COL_STATISTICS views.

**See Also:**

- "DBA_TAB_COLS"
- "USER_TAB_COLS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_TAB_COLUMNS

ALL_TAB_COLUMNS describes the columns of the tables, views, and clusters accessible to the current user. To gather statistics for this view, use the DBMS_STATS package.

### Related Views

- DBA_TAB_COLUMNS describes the columns of all tables, views, and clusters in the database.
- USER_TAB_COLUMNS describes the columns of the tables, views, and clusters owned by the current user. This view does not display the OWNER column.

This view filters out system-generated hidden columns and invisible columns, which are user-generated hidden columns. The ALL_TAB_COLS view does not filter out hidden columns and invisible columns.

**See Also:**

For more information about invisible columns:

- "ALL_TAB_COLS"
- *Oracle Database Administrator's Guide*

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table, view, or cluster |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table, view, or cluster |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name |
| DATA_TYPE | VARCHAR2(128) | | Data type of the column |
| DATA_TYPE_MOD | VARCHAR2(3) | | Data type modifier of the column |
| DATA_TYPE_OWNER | VARCHAR2(128) | | Owner of the data type of the column |
| DATA_LENGTH | NUMBER | NOT NULL | Length of the column (in bytes) |
| DATA_PRECISION | NUMBER | | Decimal precision for NUMBER data type; binary precision for FLOAT data type; NULL for all other data types |
| DATA_SCALE | NUMBER | | Digits to the right of the decimal point in a number |
| NULLABLE | VARCHAR2(1) | | Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. The constraint should be in an ENABLE VALIDATE state. |
| COLUMN_ID | NUMBER | | Sequence number of the column as created |
| DEFAULT_LENGTH | NUMBER | | Length of the default value for the column |
| DATA_DEFAULT | LONG | | Default value for the column |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column[1] |
| HIGH_VALUE | RAW(1000) | | High value in the column[1] |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | If a histogram is not available on COLUMN_NAME, then the value of this column is 1/ NUM_DISTINCT.[1] |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| NUM_BUCKETS | NUMBER | | Number of buckets in the histogram for the column<br>**Note:** The number of buckets in a histogram is specified in the SIZE parameter of the ANALYZE SQL statement. However, Oracle Database does not create a histogram with more buckets than the number of rows in the sample. Also, if the sample contains any values that are very repetitious, Oracle Database creates the specified number of buckets, but the value indicated by this column may be smaller because of an internal compression algorithm. |
| LAST_ANALYZED | DATE | | Date on which this column was most recently analyzed |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this column |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Name of the character set:<br>• CHAR_CS<br>• NCHAR_CS |
| CHAR_COL_DECL_LENGTH | NUMBER | | Declaration length of the character type column |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| CHAR_LENGTH | NUMBER | | Displays the length of the column in characters. This value only applies to the following datatypes:<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| CHAR_USED | VARCHAR2(1) | | Indicates that the column uses BYTE length semantics (B) or CHAR length semantics (C), or whether the datatype is not any of the following (NULL):<br>• CHAR<br>• VARCHAR2 |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NCHAR<br>• NVARCHAR2 |
| V80_FMT_IMAGE | VARCHAR2(3) | | Indicates whether the column data is in release 8.0 image format (YES) or not (NO) |
| DATA_UPGRADED | VARCHAR2(3) | | Indicates whether the column data has been upgraded to the latest type version format (YES) or not (NO) |
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE<br>• FREQUENCY<br>• TOP-FREQUENCY<br>• HEIGHT BALANCED<br>• HYBRID |
| DEFAULT_ON_NULL | VARCHAR2(3) | | Indicates whether the column has DEFAULT ON NULL semantics (YES) or not (NO) |
| IDENTITY_COLUMN | VARCHAR2(3) | | Indicates whether this is an identity column (YES) or not (NO) |
| EVALUATION_EDITION | VARCHAR2(128) | | Name of the edition in which editioned objects referenced in an expression column are resolved |
| UNUSABLE_BEFORE | VARCHAR2(128) | | Name of the oldest edition in which the column is usable |
| UNUSABLE_BEGINNING | VARCHAR2(128) | | Name of the oldest edition in which the column becomes perpetually unusable |

[1] These columns remain for backward compatibility with Oracle7. This information is now in the {TAB | PART}_COL_STATISTICS views.

**See Also:**

- "DBA_TAB_COLUMNS"
- "USER_TAB_COLUMNS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_TAB_COMMENTS

ALL_TAB_COMMENTS displays comments on the tables and views accessible to the current user.

### Related Views

- DBA_TAB_COMMENTS displays comments on all tables and views in the database.
- USER_TAB_COMMENTS displays comments on the tables and views owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| TABLE_TYPE | VARCHAR2(11) | | Type of the object |
| COMMENTS | VARCHAR2(4000) | | Comment on the object |
| ORIGIN_CON_ID | NUMBER | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs<br>• *n*: This value is used for rows containing data that originate in the container with the ID *n* (*n*=1 if the data originates in root) |

**See Also:**

- "DBA_TAB_COMMENTS"
- "USER_TAB_COMMENTS"

# ALL_TAB_HISTGRM_PENDING_STATS

ALL_TAB_HISTGRM_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions accessible to the current user.

### Related Views

- DBA_TAB_HISTGRM_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions in the database.
- USER_TAB_HISTGRM_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |
| ENDPOINT_NUMBER | NUMBER | | Endpoint number |
| ENDPOINT_VALUE | NUMBER | | Normalized endpoint value |
| ENDPOINT_ACTUAL_VALUE | VARCHAR2(4000) | | Actual endpoint value |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ENDPOINT_ACTUAL_VALUE_RAW | RAW(1000) | | Endpoint actual value in raw format |
| ENDPOINT_REPEAT_COUNT | NUMBER | | Frequency of the endpoint (applies only to hybrid histograms, and is set to 0 for other histogram types) |

**See Also:**

- "DBA_TAB_HISTGRM_PENDING_STATS"
- "USER_TAB_HISTGRM_PENDING_STATS"

# ALL_TAB_HISTOGRAMS

ALL_TAB_HISTOGRAMS describes histograms on tables and views accessible to the current user.

### Related Views

- DBA_TAB_HISTOGRAMS describes histograms on all tables and views in the database.
- USER_TAB_HISTOGRAMS describes histograms on all tables and views owned by the current user. This view does not display the OWNER column.

**Note:**

These views are populated only if you collect statistics on the table using the DBMS_STATS package. For more information, see *Oracle Database PL/SQL Packages and Types Reference*.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| COLUMN_NAME | VARCHAR2(4000) | | Column name or attribute of the object type column |
| ENDPOINT_NUMBER | NUMBER | | Histogram bucket number |
| ENDPOINT_VALUE | NUMBER | | Normalized endpoint value for this bucket |
| ENDPOINT_ACTUAL_VALUE | VARCHAR2(4000) | | Actual (not normalized) string value of the endpoint for this bucket |
| ENDPOINT_ACTUAL_VALUE_RAW | RAW(1000) | | Endpoint actual value in raw format |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ENDPOINT_REPEAT_COUNT | NUMBER | | Frequency of the endpoint (applies only to hybrid histograms, and is set to 0 for other histogram types) |
| SCOPE | VARCHAR2(7) | | The value is SHARED for statistics gathered on any table other than global temporary tables.<br>For a global tempoary table, the possible values are:<br>• SESSION - Indicates that the statistics are session-specific<br>• SHARED - Indicates that the statistics are shared across all sessions<br>See *Oracle Database PL/SQL Packages and Types Reference* for information about using the GLOBAL_TEMP_TABLE_STATS preference of the DBMS_STATS package to control whether to gather session or shared statistics for global temporary tables. |

**See Also:**

- "DBA_TAB_HISTOGRAMS"
- "USER_TAB_HISTOGRAMS"

# ALL_TAB_IDENTITY_COLS

ALL_TAB_IDENTITY_COLS describes all table identity columns.

### Related Views

- DBA_TAB_IDENTITY_COLS describes all table identity columns.
- USER_TAB_IDENTITY_COLS describes all table identity columns. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the identity column |
| GENERATION_TYPE | VARCHAR2(10) | | Generation type of the identity column. Possible values are ALWAYS or BY DEFAULT. |
| SEQUENCE_NAME[1] | VARCHAR2(128) | NOT NULL | Name of the sequence associated with the identity column |
| IDENTITY_OPTIONS | VARCHAR2(298) | | Options for the identity column sequence generator |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_TAB_IDENTITY_COLS"
- "USER_TAB_IDENTITY_COLS"

**See Also:**

- The ALTER TABLE statement in *Oracle Database SQL Language Reference* for more information about creating an identity column
- The CREATE TABLE statements in *Oracle Database SQL Language Reference* for more information about creating an identity column
- *Oracle Database Migration Guide* for more information about migrating third-party databases with identity columns to Oracle Database

# ALL_TAB_MODIFICATIONS

ALL_TAB_MODIFICATIONS describes tables accessible to the current user that have been modified since the last time statistics were gathered on the tables.

### Related Views

- DBA_TAB_MODIFICATIONS describes such information for all tables in the database.
- USER_TAB_MODIFICATIONS describes such information for tables owned by the current user. This view does not display the TABLE_OWNER column.

**Note:**

These views are populated only for tables with the MONITORING attribute. They are intended for statistics collection over a long period of time. For performance reasons, the Oracle Database does not populate these views immediately when the actual modifications occur. Run the FLUSH_DATABASE_MONITORING_INFO procedure in the DBMS_STATS PL/SQL package to populate these views with the latest information. The ANALYZE_ANY system privilege is required to run this procedure.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | | Owner of the modified table |
| TABLE_NAME | VARCHAR2(128) | | Name of the modified table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the modified partition |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the modified subpartition |
| INSERTS | NUMBER | | Approximate number of inserts since the last time statistics were gathered |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| UPDATES | NUMBER | | Approximate number of updates since the last time statistics were gathered |
| DELETES | NUMBER | | Approximate number of deletes since the last time statistics were gathered |
| TIMESTAMP | DATE | | Indicates the last time the table was modified |
| TRUNCATED | VARCHAR2(3) | | Indicates whether the table has been truncated since the last analyze (YES) or not (NO) |
| DROP_SEGMENTS | NUMBER | | Number of partition and subpartition segments dropped since the last analyze |

**See Also:**

- "DBA_TAB_MODIFICATIONS"
- "USER_TAB_MODIFICATIONS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO procedure

# ALL_TAB_PARTITIONS

ALL_TAB_PARTITIONS displays partition-level partitioning information, partition storage parameters, and partition statistics generated by the DBMS_STATS package for the partitions accessible to the current user.

### Related Views

- DBA_TAB_PARTITIONS displays such information for all partitions in the database.
- USER_TAB_PARTITIONS displays such information for the partitions of all partitioned objects owned by the current user. This view does not display the TABLE_OWNER column.

**Note:**

Columns marked with an asterisk (*) are populated only if you collect statistics on the table with the DBMS_STATS package.

**Note:**

The following is true for the columns below that include double asterisks (**) in the column description:

The column can display information about segment-level attributes (for simple partitioned tables) or metadata (for composite partitioned tables). In a simple partitioned table, the partition physically contains the data (the segment) in the database. In a composite partitioned table, the partition is metadata and the data itself is stored in the subpartitions.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| COMPOSITE | VARCHAR2(3) | | Indicates whether the table is composite-partitioned (YES) or not (NO) |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_COUNT | NUMBER | | If this is a composite partitioned table, the number of subpartitions in the partition |
| HIGH_VALUE | LONG | | Partition bound value expression |
| HIGH_VALUE_LENGTH | NUMBER | | Length of the partition bound value expression |
| PARTITION_POSITION | NUMBER | | Position of the partition within the table |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the partition** |
| PCT_FREE | NUMBER | | Minimum percentage of free space in a block** |
| PCT_USED | NUMBER | | Minimum percentage of used space in a block** |
| INI_TRANS | NUMBER | | Initial number of transactions** |
| MAX_TRANS | NUMBER | | Maximum number of transactions** |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent in bytes (for a range partition); size of the initial extent in blocks (for a composite partition)** |
| NEXT_EXTENT | NUMBER | | Size of secondary extents in bytes (for a range partition); size of secondary extents in blocks (for a composite partition)** |
| MIN_EXTENT | NUMBER | | Minimum number of extents allowed in the segment** |
| MAX_EXTENT | NUMBER | | Maximum number of extents allowed in the segment** |
| MAX_SIZE | NUMBER | | Maximum number of blocks allowed in the segment** |
| PCT_INCREASE | NUMBER | | Percentage increase in extent size** |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FREELISTS | NUMBER | | Number of process freelists allocated in this segment** |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated in this segment** |
| LOGGING | VARCHAR2(7) | | Indicates whether or not changes to the table are logged:**<br>• NONE - Not specified<br>**See Also:** the *_TAB_SUBPARTITIONS view<br>• YES<br>• NO |
| COMPRESSION | VARCHAR2(8) | | Indicates the actual compression property for a partition of a simple partitioned table, or the default (if specified) for subpartitions for composite partitioned tables, otherwise NONE.**<br>• NONE - The partition is composite, and a default setting is not specified for compression.<br>**See Also:** the *_TAB_SUBPARTITIONS view<br>• ENABLED - The setting for compression is enabled.<br>• DISABLED - The setting for compression is disabled. |
| COMPRESS_FOR | VARCHAR2(30) | | Default compression for what kind of operations:**<br>• BASIC<br>• ADVANCED<br>• QUERY LOW<br>• QUERY HIGH[1]<br>• ARCHIVE LOW[1]<br>• ARCHIVE HIGH[1]<br>• NULL |
| NUM_ROWS* | NUMBER | | Number of rows in the partition |
| BLOCKS* | NUMBER | | Number of used data blocks in the partition |
| EMPTY_BLOCKS | NUMBER | | Number of empty (never used) data blocks in the partition. This column is populated only if you collect statistics on the table using the DBMS_STATS package. |
| AVG_SPACE* | NUMBER | | Average amount of free space, in bytes, in a data block allocated to the partition |
| CHAIN_CNT* | NUMBER | | Number of rows in the partition that are chained from one data block to another, or which have migrated to a new block, requiring a link to preserve the old ROWID |
| AVG_ROW_LEN* | NUMBER | | Average length of a row in the partition (in bytes) |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this partition |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_ANALYZED | DATE | | Date on which this partition was most recently analyzed |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool to be used for the partition blocks:**<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for partition blocks:**<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for partition blocks:**<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the partition as a whole (YES) or they were not gathered (NO). Note that partition statistics are estimated from underlying subpartition statistics when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| IS_NESTED | VARCHAR2(3) | | Indicates whether this is a nested table partition (YES) or not (NO)<br>**See Also:** the *_NESTED_TABLES view for the parent table name/column |
| PARENT_TABLE_PARTITION | VARCHAR2(128) | | Parent table's corresponding partition<br>**See Also:** the *_NESTED_TABLES view for the parent table name/column |
| INTERVAL | VARCHAR2(3) | | Indicates whether the partition is in the interval section of an interval partitioned table (YES) or whether the partition is in the range section (NO) |
| SEGMENT_CREATED | VARCHAR2(4) | | Indicates the actual segment creation property for a partition of a simple partitioned table, or the default (if specified) for subpartitions for composite partitioned tables, otherwise NONE.**<br>For a simple partitioned table, this column indicates whether a segment was created (YES) or not (NO).<br>For composite partitioned tables, this column indicates whether or not a default segment |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | creation property was explicitly specified. Possible values:<br>• NONE - No default segment creation property was specified at the partition level. This value appears only for composite partitions, and is treated as an unspecified value.<br>• YES - Immediate segment creation was explicitly specified at the partition level and will be used as the default for all of its subpartitions.<br>• NO - Deferred segment creation was explicitly specified at the partition level and will be used as the default for all of its subpartitions. |
| INDEXING | VARCHAR2(4) | | Indicates the actual indexing property for a partition of a simple partitioned table, or the default (if specified) for subpartitions for composite partitioned tables, otherwise NONE.**<br>Possible values:<br>• NONE - The partition is composite, and a default setting is not specified for indexing.<br>This value appears only for composite partitions, and is treated as an unspecified value. When a user adds a subpartition to a table, since the defaults for the partition are unspecified, the ALL_PART_TABLES.DEF_INDEXING value is used for the newly created subpartition.<br>• ON - INDEXING is on.<br>• OFF - INDEXING is off. |
| INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is enabled (ENABLED) or disabled (DISABLED) for this partition |
| INMEMORY_PRIORITY[2] | VARCHAR2(8) | | Indicates the priority for In-Memory Column Store (IM column store) population. Possible values:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| INMEMORY_DISTRIBUTE[2] | VARCHAR2(15) | | Indicates how the IM column store is distributed in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| INMEMORY_COMPRESSION[2] | VARCHAR2(17) | | Indicates the compression level for the IM column store:<br>• NO MEMCOMPRESS |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ]<br>• NULL[3] |
| INMEMORY_DUPLICATE[2] | VARCHAR2(13) | | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |

1 Hybrid Columnar Compression is a feature of the Enterprise Edition of Oracle Database that is dependent on the underlying storage system. See *Oracle Database Concepts* for more information.

2 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

3 This column has a value based on where the segments lie for a table. For example, if the table is partitioned and is enabled for the IM column store, the value is NULL for ALL_TABLES but non-NULL for ALL_TAB_PARTITIONS.

**See Also:**

- "DBA_TAB_PARTITIONS"
- "USER_TAB_PARTITIONS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_TAB_PENDING_STATS

ALL_TAB_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions accessible to the current user.

### Related Views

- DBA_TAB_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions in the database.
- USER_TAB_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| NUM_ROWS | NUMBER | | Number of rows |
| BLOCKS | NUMBER | | Number of blocks |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AVG_ROW_LEN | NUMBER | | Average row length |
| SAMPLE_SIZE | NUMBER | | Sample size |
| LAST_ANALYZED | DATE | | Time of last analyze operation |

**See Also:**

- "DBA_TAB_PENDING_STATS"
- "USER_TAB_PENDING_STATS"

# ALL_TAB_PRIVS

ALL_TAB_PRIVS describes the following types of grants:

- Object grants for which the current user is the object owner, grantor, or grantee
- Object grants for which an enabled role or PUBLIC is the grantee

### Related Views

- DBA_TAB_PRIVS describes all object grants in the database.
- USER_TAB_PRIVS describes the object grants for which the current user is the object owner, grantor, or grantee.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| TABLE_SCHEMA | VARCHAR2(128) | | Schema of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the object |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |
| HIERARCHY | VARCHAR2(3) | | Indicates whether the privilege was granted with the HIERARCHY OPTION (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |
| TYPE | VARCHAR2(24) | | Type of the object |

**See Also:**

- "DBA_TAB_PRIVS"
- "USER_TAB_PRIVS"

# ALL_TAB_PRIVS_MADE

ALL_TAB_PRIVS_MADE describes the object grants for which the current user is the object owner or grantor.

### Related View

USER_TAB_PRIVS_MADE describes the object grants for which the current user is the object owner. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the object |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |
| HIERARCHY | VARCHAR2(3) | | Indicates whether the privilege was granted with the HIERARCHY OPTION (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |
| TYPE | VARCHAR2(24) | | Type of the object |

**See Also:**

"USER_TAB_PRIVS_MADE"

# ALL_TAB_PRIVS_RECD

ALL_TAB_PRIVS_RECD describes the following types of grants:

- Object grants for which the current user is the grantee
- Object grants for which an enabled role or PUBLIC is the grantee

### Related View

USER_TAB_PRIVS_RECD describes the object grants for which the current user is the grantee. This view does not display the GRANTEE column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the object |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |
| HIERARCHY | VARCHAR2(3) | | Indicates whether the privilege was granted with the HIERARCHY OPTION (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |
| TYPE | VARCHAR2(24) | | Type of the object |

**See Also:**

"USER_TAB_PRIVS_RECD"

# ALL_TAB_STAT_PREFS

ALL_TAB_STAT_PREFS displays information about statistics preferences for the tables accessible to the current user.

### Related Views

- DBA_TAB_STAT_PREFS displays information about statistics preferences for all tables in the database.
- USER_TAB_STAT_PREFS displays information about statistics preferences for the tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| PREFERENCE_NAME | VARCHAR2(30) | | Name of the preference |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PREFERENCE_VALUE | VARCHAR2(4000) | | Value of the preference |

**See Also:**

- "DBA_TAB_STAT_PREFS"
- "USER_TAB_STAT_PREFS"

# ALL_TAB_STATISTICS

ALL_TAB_STATISTICS displays optimizer statistics for the tables accessible to the current user.

### Related Views

- DBA_TAB_STATISTICS displays optimizer statistics for all tables in the database.
- USER_TAB_STATISTICS displays optimizer statistics for the tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| PARTITION_POSITION | NUMBER | | Position of the partition within the table |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| SUBPARTITION_POSITION | NUMBER | | Position of the subpartition within the partition |
| OBJECT_TYPE | VARCHAR2(12) | | Type of the object:<br>• TABLE<br>• PARTITION<br>• SUBPARTITION |
| NUM_ROWS | NUMBER | | Number of rows in the object |
| BLOCKS | NUMBER | | Number of used blocks in the object |
| EMPTY_BLOCKS | NUMBER | | Number of empty blocks in the object |
| AVG_SPACE | NUMBER | | Average available free space in the object |
| CHAIN_CNT | NUMBER | | Number of chained rows in the object |
| AVG_ROW_LEN | NUMBER | | Average row length, including row overhead |
| AVG_SPACE_FREELIST_BLOCKS | NUMBER | | Average freespace of all blocks on a freelist |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NUM_FREELIST_BLOCKS | NUMBER | | Number of blocks on the freelist |
| AVG_CACHED_BLOCKS | NUMBER | | Average number of blocks in the buffer cache |
| AVG_CACHE_HIT_RATIO | NUMBER | | Average cache hit ratio for the object |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing the table |
| LAST_ANALYZED | DATE | | Date of the most recent time the table was analyzed |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| STATTYPE_LOCKED | VARCHAR2(5) | | Type of statistics lock:<br>• DATA<br>• CACHE<br>• ALL |
| STALE_STATS | VARCHAR2(3) | | Indicates whether statistics for the object are stale (YES) or not (NO) |
| SCOPE | VARCHAR2(7) | | The value is SHARED for statistics gathered on any table other than global temporary tables.<br>For a global tempoary table, the possible values are:<br>• SESSION - Indicates that the statistics are session-specific<br>• SHARED - Indicates that the statistics are shared across all sessions<br>See *Oracle Database PL/SQL Packages and Types Reference* for information about using the GLOBAL_TEMP_TABLE_STATS preference of the DBMS_STATS package to control whether to gather session or shared statistics for global temporary tables. |

**See Also:**

- "DBA_TAB_STATISTICS"
- "USER_TAB_STATISTICS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_TAB_STATS_HISTORY

ALL_TAB_STATS_HISTORY provides a history of table statistics modifications for all tables accessible to the current user.

### Related Views

- DBA_TAB_STATS_HISTORY provides a history of table statistics modifications for all tables in the database.
- USER_TAB_STATS_HISTORY provides a history of table statistics modifications for all tables owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| STATS_UPDATE_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time at which the statistics were updated |

**See Also:**

- "DBA_TAB_STATS_HISTORY"
- "USER_TAB_STATS_HISTORY"

# ALL_TAB_SUBPARTITIONS

ALL_TAB_SUBPARTITIONS displays, for each table subpartition accessible to the current user, the subpartition name, name of the table and partition to which it belongs, its storage attributes, and statistics generated by the DBMS_STATS package.

### Related Views

- DBA_TAB_SUBPARTITIONS displays such information for all subpartitions in the database.
- USER_TAB_SUBPARTITIONS displays such information for subpartitions of all partitioned objects owned by the current user. This view does not display the TABLE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SUBPARTITION_NAME | VARCHAR2(128) | | Name of the subpartition |
| HIGH_VALUE | LONG | | Subpartition bound value expression |
| HIGH_VALUE_LENGTH | NUMBER | NOT NULL | Length of the subpartition bound value expression |
| SUBPARTITION_POSITION | NUMBER | | Position of the subpartition within the partition |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace containing the subpartition |
| PCT_FREE | NUMBER | NOT NULL | Minimum percentage of free space in a block |
| PCT_USED | NUMBER | | Minimum percentage of used space in a block |
| INI_TRANS | NUMBER | NOT NULL | Initial number of transactions |
| MAX_TRANS | NUMBER | NOT NULL | Maximum number of transactions |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent in bytes (for a range partition); size of the initial extent in blocks (for a composite partition) |
| NEXT_EXTENT | NUMBER | | Size of secondary extents in bytes (for a range partition); size of secondary extents in blocks (for a composite partition) |
| MIN_EXTENT | NUMBER | | Minimum number of extents allowed in the segment |
| MAX_EXTENT | NUMBER | | Maximum number of extents allowed in the segment |
| MAX_SIZE | NUMBER | | Maximum number of blocks allowed in the segment |
| PCT_INCREASE | NUMBER | | Percentage increase in extent size |
| FREELISTS | NUMBER | | Number of freelist groups allocated in this segment |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated in this segment |
| LOGGING | VARCHAR2(3) | | Indicates whether or not changes to the table are logged:<br>• YES<br>• NO |
| COMPRESSION | VARCHAR2(8) | | Indicates whether this subpartition is compressed (ENABLED) or not (DISABLED) |
| COMPRESS_FOR | VARCHAR2(30) | | Default compression for what kind of operations:<br>• BASIC<br>• ADVANCED<br>• QUERY LOW<br>• QUERY HIGH[1] |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • ARCHIVE LOW[1]<br>• ARCHIVE HIGH[1]<br>• NULL |
| NUM_ROWS | NUMBER | | Number of rows in the subpartition |
| BLOCKS | NUMBER | | Number of blocks in the subpartition |
| EMPTY_BLOCKS | NUMBER | | Number of empty blocks in the subpartition |
| AVG_SPACE | NUMBER | | Average space in the subpartition |
| CHAIN_CNT | NUMBER | | Chain count |
| AVG_ROW_LEN | NUMBER | | Average row length |
| SAMPLE_SIZE | NUMBER | | Sample size |
| LAST_ANALYZED | DATE | | Date on which this table was most recently analyzed |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool for this subpartition:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for subpartition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for subpartition blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered on a subpartition (YES) or they were not gathered (NO) |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| INTERVAL | VARCHAR2(3) | | Indicates whether the partition is in the interval section of an interval partitioned table (YES) or whether the partition is in the range section (NO) |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the table subpartition segment has been created (YES) or not (NO); N/A indicates that this table is not subpartitioned |
| INDEXING | VARCHAR2(3) | | Indicates the indexing property.<br>Possible values: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • ON - Indexing is on for this subpartition<br>• OFF - Indexing is off for this subpartition |
| INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is enabled (ENABLED) or disabled (DISABLED) for this subpartition |
| INMEMORY_PRIORITY[2] | VARCHAR2(8) | | Indicates the priority for In-Memory Column Store (IM column store) population. Possible values:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| INMEMORY_DISTRIBUTE[2] | VARCHAR2(15) | | Indicates how the IM column store is distributed in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• DBY PARTITION<br>• BY SUBPARTITION |
| INMEMORY_COMPRESSION[2] | VARCHAR2(17) | | Indicates the compression level for the IM column store in an Oracle RAC environment:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ]<br>• NULL[3] |
| INMEMORY_DUPLICATE[2] | | | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• DUPLICATE<br>• NO DUPLICATE<br>• DUPLICATE ALL |

1 Hybrid Columnar Compression is a feature of the Enterprise Edition of Oracle Database that is dependent on the underlying storage system. See *Oracle Database Concepts* for more information.

2 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

3 This column has a value based on where the segments lie for a table. For example, if the table is partitioned and is enabled for IM column store, the value is NULL for ALL TABLES but non-NULL for ALL_TAB_SUBPARTITIONS.

**See Also:**

- "DBA_TAB_SUBPARTITIONS"
- "USER_TAB_SUBPARTITIONS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_TABLES

ALL_TABLES describes the relational tables accessible to the current user. To gather statistics for this view, use the DBMS_STATS package.

### Related Views

- DBA_TABLES describes all relational tables in the database.
- USER_TABLES describes the relational tables owned by the current user. This view does not display the OWNER column.

**Note:**

Columns marked with an asterisk (*) are populated only if you collect statistics on the table with the DBMS_STATS package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the table; NULL for partitioned, temporary, and index-organized tables |
| CLUSTER_NAME | VARCHAR2(128) | | Name of the cluster, if any, to which the table belongs |
| IOT_NAME | VARCHAR2(128) | | Name of the index-organized table, if any, to which the overflow or mapping table entry belongs. If the IOT_TYPE column is not NULL, then this column contains the base table name. |
| STATUS | VARCHAR2(8) | | If a previous DROP TABLE operation failed, indicates whether the table is unusable (UNUSABLE) or valid (VALID) |
| PCT_FREE | NUMBER | | Minimum percentage of free space in a block; NULL for partitioned tables |
| PCT_USED | NUMBER | | Minimum percentage of used space in a block; NULL for partitioned tables |
| INI_TRANS | NUMBER | | Initial number of transactions; NULL for partitioned tables |
| MAX_TRANS | NUMBER | | Maximum number of transactions; NULL for partitioned tables |
| INITIAL_EXTENT | NUMBER | | Size of the initial extent (in bytes); NULL for partitioned tables |
| NEXT_EXTENT | NUMBER | | Size of secondary extents (in bytes); NULL for partitioned tables |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MIN_EXTENTS | NUMBER | | Minimum number of extents allowed in the segment; NULL for partitioned tables |
| MAX_EXTENTS | NUMBER | | Maximum number of extents allowed in the segment; NULL for partitioned tables |
| PCT_INCREASE | NUMBER | | Percentage increase in extent size; NULL for partitioned tables |
| FREELISTS | NUMBER | | Number of process freelists allocated to the segment; NULL for partitioned tables |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated to the segment; NULL for partitioned tables |
| LOGGING | VARCHAR2(3) | | Indicates whether or not changes to the table are logged; NULL for partitioned tables:<br>• YES<br>• NO |
| BACKED_UP | VARCHAR2(1) | | Indicates whether the table has been backed up since the last modification (Y) or not (N) |
| NUM_ROWS* | NUMBER | | Number of rows in the table |
| BLOCKS* | NUMBER | | Number of used data blocks in the table |
| EMPTY_BLOCKS | NUMBER | | Number of empty (never used) data blocks in the table. This column is populated only if you collect statistics on the table using the DBMS_STATS package. |
| AVG_SPACE* | NUMBER | | Average amount of free space, in bytes, in a data block allocated to the table |
| CHAIN_CNT* | NUMBER | | Number of rows in the table that are chained from one data block to another, or which have migrated to a new block, requiring a link to preserve the old ROWID |
| AVG_ROW_LEN* | NUMBER | | Average length of a row in the table (in bytes) |
| AVG_SPACE_FREELIST_BLOCKS | NUMBER | | Average freespace of all blocks on a freelist |
| NUM_FREELIST_BLOCKS | NUMBER | | Number of blocks on the freelist |
| DEGREE | VARCHAR2(10) | | Number of threads per instance for scanning the table, or DEFAULT |
| INSTANCES | VARCHAR2(10) | | Number of instances across which the table is to be scanned, or DEFAULT |
| CACHE | VARCHAR2(5) | | Indicates whether the table is to be cached in the buffer cache (Y) or not (N) |
| TABLE_LOCK | VARCHAR2(8) | | Indicates whether table locking is enabled (ENABLED) or disabled (DISABLED) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this table |
| LAST_ANALYZED | DATE | | Date on which this table was most recently analyzed |
| PARTITIONED | VARCHAR2(3) | | Indicates whether the table is partitioned (YES) or not (NO) |
| IOT_TYPE | VARCHAR2(12) | | If the table is an index-organized table, then IOT_TYPE is IOT, IOT_OVERFLOW, or IOT_MAPPING. If the table is not an index-organized table, then IOT_TYPE is NULL. |
| TEMPORARY | VARCHAR2(1) | | Indicates whether the table is temporary (Y) or not (N) |
| SECONDARY | VARCHAR2(1) | | Indicates whether the table is a secondary object created by the ODCIIndexCreate method of the Oracle Data Cartridge (Y) or not (N) |
| NESTED | VARCHAR2(3) | | Indicates whether the table is a nested table (YES) or not (NO) |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool for the table; NULL for partitioned tables:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>• NULL |
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for table blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for table blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| ROW_MOVEMENT | VARCHAR2(8) | | Indicates whether partitioned row movement is enabled (ENABLED) or disabled (DISABLED) |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DURATION | VARCHAR2(15) | | Indicates the duration of a temporary table:<br>• SYS$SESSION - Rows are preserved for the duration of the session<br>• SYS$TRANSACTION - Rows are deleted after COMMIT<br>Null - Permanent table |
| SKIP_CORRUPT | VARCHAR2(8) | | Indicates whether Oracle Database ignores blocks marked corrupt during table and index scans (ENABLED) or raises an error (DISABLED). To enable this feature, run the DBMS_REPAIR.SKIP_CORRUPT_BLOCKS procedure. |
| MONITORING | VARCHAR2(3) | | Indicates whether the table has the MONITORING attribute set (YES) or not (NO) |
| CLUSTER_OWNER | VARCHAR2(128) | | Owner of the cluster, if any, to which the table belongs |
| DEPENDENCIES | VARCHAR2(8) | | Indicates whether row-level dependency tracking is enabled (ENABLED) or disabled (DISABLED) |
| COMPRESSION | VARCHAR2(8) | | Indicates whether table compression is enabled (ENABLED) or not (DISABLED); NULL for partitioned tables |
| COMPRESS_FOR | VARCHAR2(30) | | Default compression for what kind of operations:<br>• BASIC<br>• ADVANCED<br>• QUERY LOW<br>• QUERY HIGH[1]<br>• ARCHIVE LOW[1]<br>• ARCHIVE HIGH[1]<br>• QUERY LOW ROW LEVEL LOCKING[1]<br>• QUERY HIGH ROW LEVEL LOCKING[1]<br>• ARCHIVE LOW ROW LEVEL LOCKING[1]<br>• ARCHIVE HIGH ROW LEVEL LOCKING[1]<br>• NO ROW LEVEL LOCKING[1]<br>• NULL |
| DROPPED | VARCHAR2(3) | | Indicates whether the table has been dropped and is in the recycle bin (YES) or not (NO); NULL for partitioned tables |
| READ_ONLY | VARCHAR2(3) | | Indicates whether the table IS READ-ONLY (YES) or not (NO) |
| SEGMENT_CREATED | VARCHAR2(3) | | Indicates whether the table segment is created (YES) or not (NO) |
| RESULT_CACHE | VARCHAR2(7) | | Result cache mode annotation for the table:<br>• DEFAULT - Table has not been annotated<br>• FORCE<br>• MANUAL |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLUSTERING | VARCHAR2(3) | | Indicates whether the table has the attribute clustering clause (YES) or not (NO) |
| ACTIVITY_TRACKING | VARCHAR2(23) | | Indicates whether Heat Map tracking is enabled on the table |
| DML_TIMESTAMP | VARCHAR2(25) | | Modification time, creation time, or both for Automatic Data Optimization |
| HAS_IDENTITY | VARCHAR2(3) | | Indicates whether the table has an identity column (YES) or not (NO) |
| CONTAINER_DATA | VARCHAR2(3) | | Indicates whether the table contains container-specific data. Possible values:<br>• YES if the table was created with the CONTAINER_DATA clause<br>• NO otherwise |
| INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is enabled (ENABLED) or disabled (DISABLED) for this table |
| INMEMORY_PRIORITY[2] | VARCHAR2(8) | | Indicates the priority for In-Memory Column Store (IM column store) population. Possible values:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| INMEMORY_DISTRIBUTE[2] | VARCHAR2(15) | | Indicates how the IM column store is distributed in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• DUPLICATE<br>• NONE<br>• NULL[3] |
| INMEMORY_COMPRESSION[2] | VARCHAR2(17) | | Indicates the compression level for the IM column store:<br>• BASIC<br>• FOR CAPACITY [ HIGH \| LOW ]<br>• FOR CAPACITY QUERY<br>• NULL[3] |
| INMEMORY_DUPLICATE[2] | VARCHAR2(13) | | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |

[1] Hybrid Columnar Compression is a feature of the Enterprise Edition of Oracle Database that is dependent on the underlying storage system. See *Oracle Database Concepts* for more information..
[2] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

[3] This column has a value based on where the segments lie for a table. For example, if the table is partitioned and is enabled for the IM column store, the value is NULL for ALL_TABLES but non-NULL for ALL_TAB_PARTITIONS.

**See Also:**

- "DBA_TABLES"
- "USER_TABLES"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# ALL_TRANSFORMATIONS

ALL_TRANSFORMATIONS displays information about all transformations accessible to the current user. These transformations can be specified with Advanced Queuing operations such as enqueue, dequeue, and subscribe to automatically integrate transformations in AQ messaging.

### Related Views

- DBA_TRANSFORMATIONS displays information about all transformations in the database.
- USER_TRANSFORMATIONS displays information about transformations owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRANSFORMATION_ID | NUMBER | NOT NULL | Unique identifier for the transformation |
| OWNER | VARCHAR2(128) | NOT NULL | Owning user of the transformation |
| NAME | VARCHAR2(128) | NOT NULL | Transformation name |
| FROM_TYPE | VARCHAR2(128) | | Source type name |
| TO_TYPE | VARCHAR2(256) | | Target type name |

**See Also:**

- "DBA_TRANSFORMATIONS"
- "USER_TRANSFORMATIONS"

# ALL_TRIGGER_COLS

ALL_TRIGGER_COLS describes the use of columns in the triggers accessible to the current user and in triggers on tables accessible to the current user. If the user has the CREATE ANY TRIGGER privilege, then this view describes the use of columns in all triggers in the database.

### Related Views

- DBA_TRIGGER_COLS describes the use of columns in all triggers in the database.
- USER_TRIGGER_COLS describes the use of columns in the triggers owned by the current user and in triggers on tables owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRIGGER_OWNER | VARCHAR2(128) | | Owner of the trigger |
| TRIGGER_NAME | VARCHAR2(128) | | Name of the trigger |
| TABLE_OWNER | VARCHAR2(128) | | Owner of the table on which the trigger is defined |
| TABLE_NAME | VARCHAR2(128) | | Table on which the trigger is defined |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column used in the trigger |
| COLUMN_LIST | VARCHAR2(3) | | Indicates whether the column is specified in the UPDATE clause (YES) or not (NO) |
| COLUMN_USAGE | VARCHAR2(17) | | How the column is used in the trigger:<br>• NEW IN<br>• OLD IN<br>• NEW IN OLD IN<br>• NEW OUT<br>• NEW IN OUT<br>• NEW OUT OLD IN<br>• NEW IN OUT OLD IN<br>• PARENT IN |

**See Also:**

- "DBA_TRIGGER_COLS"
- "USER_TRIGGER_COLS"

## ALL_TRIGGER_ORDERING

ALL_TRIGGER_ORDERING describes the triggers accessible to the current user that have FOLLOWS or PRECEDES ordering.

### Related Views

- DBA_TRIGGER_ORDERING describes all triggers in the database that have FOLLOWS or PRECEDES ordering.
- USER_TRIGGER_ORDERING describes the triggers owned by the current user that have FOLLOWS or PRECEDES ordering. This view does not display the TRIGGER_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRIGGER_OWNER | VARCHAR2(128) | NOT NULL | Owner of the trigger |
| TRIGGER_NAME | VARCHAR2(128) | NOT NULL | Name of the trigger |
| REFERENCED_TRIGGER_OWNER | VARCHAR2(128) | | Owner of the referenced trigger |
| REFERENCED_TRIGGER_NAME | VARCHAR2(128) | | Name of the referenced trigger |
| ORDERING_TYPE | VARCHAR2(8) | | Type of the ordering between the trigger and the referenced trigger:<br>• FOLLOWS<br>• PRECEDES |

**See Also:**

- "DBA_TRIGGER_ORDERING"
- "USER_TRIGGER_ORDERING"

# ALL_TRIGGERS

ALL_TRIGGERS describes the triggers on tables accessible to the current user. If the user has the CREATE ANY TRIGGER privilege, then this view describes all triggers in the database.

### Related Views

- DBA_TRIGGERS describes all triggers in the database.
- USER_TRIGGERS describes the triggers owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the trigger |
| TRIGGER_NAME | VARCHAR2(128) | | Name of the trigger |
| TRIGGER_TYPE | VARCHAR2(16) | | When the trigger fires:<br>• BEFORE STATEMENT<br>• BEFORE EACH ROW<br>• AFTER STATEMENT<br>• AFTER EACH ROW<br>• INSTEAD OF<br>• COMPOUND |
| TRIGGERING_EVENT | VARCHAR2(246) | | DML, DDL, or database event that fires the trigger<br>**See Also:** *Oracle Database PL/SQL Language Reference* for additional information about triggers and triggering events. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | | Owner of the table on which the trigger is defined |
| BASE_OBJECT_TYPE | VARCHAR2(18) | | Base object on which the trigger is defined:<br>• TABLE<br>• VIEW<br>• SCHEMA<br>• DATABASE |
| TABLE_NAME | VARCHAR2(128) | | If the base object type of the trigger is SCHEMA or DATABASE, then this column is NULL; if the base object type of the trigger is TABLE or VIEW, then this column indicates the table or view name on which the trigger is defined |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the nested table column (if a nested table trigger), else NULL |
| REFERENCING_NAMES | VARCHAR2(422) | | Names used for referencing OLD and NEW column values from within the trigger |
| WHEN_CLAUSE | VARCHAR2(4000) | | Must evaluate to TRUE for TRIGGER_BODY to execute |
| STATUS | VARCHAR2(8) | | Indicates whether the trigger is enabled (ENABLED) or disabled (DISABLED); a disabled trigger will not fire |
| DESCRIPTION | VARCHAR2(4000) | | Trigger description; useful for re-creating a trigger creation statement |
| ACTION_TYPE | VARCHAR2(11) | | Action type of the trigger body:<br>• CALL<br>• PL/SQL |
| TRIGGER_BODY | LONG | | Statements executed by the trigger when it fires |
| CROSSEDITION | VARCHAR2(7) | | Type of crossedition trigger:<br>• FORWARD<br>• REVERSE<br>• NO |
| BEFORE_STATEMENT | VARCHAR2(3) | | Indicates whether the trigger has a BEFORE STATEMENT section (YES) or not (NO) |
| BEFORE_ROW | VARCHAR2(3) | | Indicates whether the trigger has a BEFORE EACH ROW section (YES) or not (NO) |
| AFTER_ROW | VARCHAR2(3) | | Indicates whether the trigger has an AFTER EACH ROW section (YES) or not (NO) |
| AFTER_STATEMENT | VARCHAR2(3) | | Indicates whether the trigger has an AFTER STATEMENT section (YES) or not (NO) |
| INSTEAD_OF_ROW | VARCHAR2(3) | | Indicates whether the trigger has an INSTEAD OF section (YES) or not (NO) |
| FIRE_ONCE | VARCHAR2(3) | | Indicates whether the trigger will fire only for user processes making changes (YES) or whether |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | the trigger will also fire for Streams Apply or SQL Apply processes (NO) |
| APPLY_SERVER_ONLY | VARCHAR2(3) | | Indicates whether the trigger will only fire for a Streams Apply or SQL Apply process (YES) or not (NO). If set to YES, then the setting of FIRE_ONCE does not matter.<br>**See Also:** the DBMS_DDL.SET_TRIGGER_FIRING_PROPERTY procedure |

**See Also:**

- "DBA_TRIGGERS"
- "USER_TRIGGERS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_DDL.SET_TRIGGER_FIRING_PROPERTY procedure

# ALL_TSTZ_TAB_COLS

ALL_TSTZ_TAB_COLS displays information about the columns of the tables accessible to the current user, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types.

### Related Views

- DBA_TSTZ_TAB_COLS displays information about the columns of all tables in the database, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types. This view does not display the COLUMN_NAME, NESTED, and VIRTUAL_COLUMN columns.
- USER_TSTZ_TAB_COLS displays information about the columns of the tables owned by the current user, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types. This view does not display the OWNER, COLUMN_NAME, NESTED, and VIRTUAL_COLUMN columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name |
| QUALIFIED_COL_NAME | VARCHAR2(4000) | | Qualified column name |
| NESTED | NUMBER | | Indicates whether the column is a nested table (1) or not (0) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| VIRTUAL_COLUMN | NUMBER | | Indicates whether the column is a virtual column (1) or not (0) |

**See Also:**

- "DBA_TSTZ_TAB_COLS"
- "USER_TSTZ_TAB_COLS"

# ALL_TSTZ_TABLES

ALL_TSTZ_TABLES displays information about the tables accessible to the current user, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types.

### Related Views

- DBA_TSTZ_TABLES displays information about all tables in the database, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types.
- USER_TSTZ_TABLES displays information about the tables owned by the current user, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| UPGRADE_IN_PROGRESS | VARCHAR2(3) | | Indicates whether a table upgrade is in progress (YES) or not (NO) |

**See Also:**

- "DBA_TSTZ_TABLES"
- "USER_TSTZ_TABLES"

# ALL_TYPE_ATTRS

ALL_TYPE_ATTRS describes the attributes of the object types accessible to the current user.

### Related Views

- DBA_TYPE_ATTRS describes the attributes of all object types in the database. This view does not include the CHAR_USED column.

- USER_TYPE_ATTRS describes the attributes of the object types owned by the current user. This view does not display the OWNER or CHAR_USED column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | | Name of the type |
| ATTR_NAME | VARCHAR2(128) | | Name of the attribute |
| ATTR_TYPE_MOD | VARCHAR2(7) | | Type modifier of the attribute:<br>• REF<br>• POINTER |
| ATTR_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the attribute |
| ATTR_TYPE_NAME | VARCHAR2(128) | | Name of the type of the attribute |
| LENGTH | NUMBER | | Length of the CHAR attribute, or maximum length of the VARCHAR or VARCHAR2 attribute. |
| PRECISION | NUMBER | | Decimal precision of the NUMBER or DECIMAL attribute, or binary precision of the FLOAT attribute. |
| SCALE | NUMBER | | Scale of the NUMBER or DECIMAL attribute |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Character set name of the attribute (CHAR_CS or NCHAR_CS) |
| ATTR_NO | NUMBER | | Syntactical order number or position of the attribute as specified in the type specification or CREATE TYPE statement (not to be used as an ID number) |
| INHERITED | VARCHAR2(3) | | Indicates whether the attribute is inherited from a supertype (YES) or not (NO) |
| CHAR_USED | VARCHAR2(1) | | Indicates whether the attribute uses BYTE length semantics (B) or CHAR length semantics (C). For NCHAR and NVARCHAR2 attribute types, this value is always C. |

**See Also:**

- "DBA_TYPE_ATTRS"
- "USER_TYPE_ATTRS"

# ALL_TYPE_METHODS

ALL_TYPE_METHODS describes the methods of the object types accessible to the current user.

### Related Views

- DBA_TYPE_METHODS describes the methods of all object types in the database.

- USER_TYPE_METHODS describes the methods of the object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the type |
| METHOD_NAME | VARCHAR2(128) | NOT NULL | Name of the method |
| METHOD_NO | NUMBER | NOT NULL | Method number for distinguishing overloaded methods (not to be used as ID number) |
| METHOD_TYPE | VARCHAR2(6) | | Type of the method:<br>• MAP<br>• ORDER<br>• PUBLIC |
| PARAMETERS | NUMBER | NOT NULL | Number of parameters to the method |
| RESULTS | NUMBER | NOT NULL | Number of results returned by the method |
| FINAL | VARCHAR2(3) | | Indicates whether the method is final (YES) or not (NO) |
| INSTANTIABLE | VARCHAR2(3) | | Indicates whether the method is instantiable (YES) or not (NO) |
| OVERRIDING | VARCHAR2(3) | | Indicates whether the method is overriding a supertype method (YES) or not (NO) |
| INHERITED | VARCHAR2(3) | | Indicates whether the method is inherited from a supertype (YES) or not (NO) |

**See Also:**

- "DBA_TYPE_METHODS"
- "USER_TYPE_METHODS"

# ALL_TYPE_VERSIONS

ALL_TYPE_VERSIONS describes the versions of the object types accessible to the current user.

### Related Views

- DBA_TYPE_VERSIONS describes the versions of all object types in the database.
- USER_TYPE_VERSIONS describes the versions of the object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the type |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE_NAME | VARCHAR2(128) | NOT NULL | Name of the type |
| VERSION# | NUMBER | NOT NULL | Internal version number of the type |
| TYPECODE | VARCHAR2(128) | | Typecode of the type |
| STATUS | VARCHAR2(7) | | Status of the type:<br>• N/A<br>• VALID<br>• INVALID |
| LINE | NUMBER | NOT NULL | Line number of the type's spec |
| TEXT | VARCHAR2(4000) | | Text of the type's spec |
| HASHCODE | RAW(17) | | Hashcode of the type |

**See Also:**

- "DBA_TYPE_VERSIONS"
- "USER_TYPE_VERSIONS"

# ALL_TYPES

ALL_TYPES describes the object types accessible to the current user.

### Related Views

- DBA_TYPES describes all object types in the database.
- USER_TYPES describes the object types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the type |
| TYPE_NAME | VARCHAR2(128) | | Name of the type |
| TYPE_OID | RAW(16) | | Object identifier (OID) of the type |
| TYPECODE | VARCHAR2(128) | | Typecode of the type |
| ATTRIBUTES | NUMBER | | Number of attributes (if any) in the type |
| METHODS | NUMBER | | Number of methods (if any) in the type |
| PREDEFINED | VARCHAR2(3) | | Indicates whether the type is a predefined type (YES) or not (NO) |
| INCOMPLETE | VARCHAR2(3) | | Indicates whether the type is an incomplete type (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FINAL | VARCHAR2(3) | | Indicates whether the type is a final type (YES) or not (NO) |
| INSTANTIABLE | VARCHAR2(3) | | Indicates whether the type is an instantiable type (YES) or not (NO) |
| SUPERTYPE_OWNER | VARCHAR2(128) | | Owner of the supertype (NULL if type is not a subtype) |
| SUPERTYPE_NAME | VARCHAR2(128) | | Name of the supertype (NULL if type is not a subtype) |
| LOCAL_ATTRIBUTES | NUMBER | | Number of local (not inherited) attributes (if any) in the subtype |
| LOCAL_METHODS | NUMBER | | Number of local (not inherited) methods (if any) in the subtype |
| TYPEID | RAW(16) | | Type ID value of the type |

**See Also:**

- "DBA_TYPES"
- "USER_TYPES"

# ALL_UNUSED_COL_TABS

ALL_UNUSED_COL_TABS describes the tables accessible to the current user that contain unused columns.

### Related Views

- DBA_UNUSED_COL_TABS describes all tables in the database that contain unused columns.
- USER_UNUSED_COL_TABS describes the tables owned by the current user that contain unused columns. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| COUNT | NUMBER | | Number of unused columns |

**See Also:**

- "DBA_UNUSED_COL_TABS"
- "USER_UNUSED_COL_TABS"

# ALL_UPDATABLE_COLUMNS

ALL_UPDATABLE_COLUMNS describes all columns in a join view that are updatable by the current user, subject to appropriate privileges.

### Related Views

- DBA_UPDATABLE_COLUMNS describes all columns in a join view that are updatable by the database administrator, subject to appropriate privileges.
- USER_UPDATABLE_COLUMNS describes all columns owned by the current user that are in a join view and are updatable by the current user, subject to appropriate privileges.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name |
| UPDATABLE | VARCHAR2(3) | | Indicates whether the column is updatable (YES) or not (NO) |
| INSERTABLE | VARCHAR2(3) | | Indicates whether the column is insertable (YES) or not (NO) |
| DELETABLE | VARCHAR2(3) | | Indicates whether the column is deletable (YES) or not (NO) |

**See Also:**

- "DBA_UPDATABLE_COLUMNS"
- "USER_UPDATABLE_COLUMNS"
- *Oracle Database Concepts* for information on updatable join views

# ALL_USERS

ALL_USERS lists all users of the database visible to the current user. This view does not describe the users (see the related views).

### Related Views

- DBA_USERS describes all users of the database, and contains more columns than ALL_USERS.
- USER_USERS describes the current user, and contains more columns than ALL_USERS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_ID | NUMBER | NOT NULL | ID number of the user |
| CREATED | DATE | NOT NULL | User creation date |
| COMMON | VARCHAR2(3) | | Indicates whether a given user is common. Possible values:<br>• YES if a user is common<br>• NO if a user is local (not common) |
| ORACLE_MAINTAINED | VARCHAR2(1) | | Denotes whether the user was created, and is maintained, by Oracle-supplied scripts (such as catalog.sql or catproc.sql). A user for which this column has the value Y must not be changed in any way except by running an Oracle-supplied script. |

**See Also:**

- "DBA_USERS"
- "USER_USERS"

# ALL_USTATS

ALL_USTATS describes the user-defined statistics collected on the tables and indexes accessible to the current user.

### Related Views

- DBA_USTATS describes the user-defined statistics collected on all tables and indexes in the database.
- USER_USTATS describes the user-defined statistics collected on the tables and indexes owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the table or index for which the statistics have been collected |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or index for which the statistics have been collected |
| PARTITION_NAME | VARCHAR2(128) | | Partition name of a table; NULL if the table is either nonpartitioned or the entry corresponds to the aggregate statistics for the table |
| OBJECT_TYPE | VARCHAR2(6) | | Type of the object for which statistics have been collected:<br>• INDEX<br>• COLUMN |
| ASSOCIATION | VARCHAR2(8) | | Statistics type association:<br>• DIRECT Direct association with the object for which the statistics have been collected |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • IMPLICIT - Association for which the statistics have been collected is with the column type or index type, and the object is an instance of that column type or index type |
| COLUMN_NAME | VARCHAR2(128) | | Column name, if OBJECT_TYPE is COLUMN, for which statistics have been collected |
| STATSTYPE_SCHEMA | VARCHAR2(128) | | Schema of the statistics type which was used to collect the statistics |
| STATSTYPE_NAME | VARCHAR2(128) | | Name of the statistics type which was used to collect statistics |
| STATISTICS | RAW(2000) | | User-collected statistics for the object |

**See Also:**

- "DBA_USTATS"
- "USER_USTATS"

# ALL_VARRAYS

ALL_VARRAYS describes the varrays accessible to the current user.

### Related Views

- DBA_VARRAYS describes all varrays in the database.
- USER_VARRAYS describes the varrays owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table containing the varray |
| PARENT_TABLE_NAME | VARCHAR2(128) | | Name of the containing table |
| PARENT_TABLE_COLUMN | VARCHAR2(4000) | | Name of the varray column or attribute |
| TYPE_OWNER | VARCHAR2(128) | | Owner of the varray type |
| TYPE_NAME | VARCHAR2(128) | | Name of the varray type |
| LOB_NAME | VARCHAR2(128) | | Name of the LOB if the varray is stored in a LOB |
| STORAGE_SPEC | VARCHAR2(30) | | Indicates whether the storage was defaulted (DEFAULT) or user-specified (USER_SPECIFIED) |
| RETURN_TYPE | VARCHAR2(20) | | Return type of the column:<br>• LOCATOR<br>• VALUE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ELEMENT_SUBSTITUTABLE | VARCHAR2(25) | | Indicates whether the varray element is substitutable (Y) or not (N) |

**See Also:**

- "DBA_VARRAYS"
- "USER_VARRAYS"

# ALL_VIEWS

ALL_VIEWS describes the views accessible to the current user.

### Related Views

- DBA_VIEWS describes all views in the database.
- USER_VIEWS describes the views owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the view |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the view |
| TEXT_LENGTH | NUMBER | | Length of the view text |
| TEXT | LONG | | View text. This column returns the correct value only when the row originates from the current container. The BEQUEATH clause will not appear as part of the TEXT column in this view. |
| TEXT_VC | VARCHAR2(4000) | | View text. This column may truncate the view text. The BEQUEATH clause will not appear as part of the TEXT_VC column in this view. |
| TYPE_TEXT_LENGTH | NUMBER | | Length of the type clause of the typed view |
| TYPE_TEXT | VARCHAR2(4000) | | Type clause of the typed view |
| OID_TEXT_LENGTH | NUMBER | | Length of the WITH OID clause of the typed view |
| OID_TEXT | VARCHAR2(4000) | | WITH OID clause of the typed view |
| VIEW_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the view if the view is a typed view |
| VIEW_TYPE | VARCHAR2(128) | | Type of the view if the view is a typed view |
| SUPERVIEW_NAME | VARCHAR2(128) | | Name of the superview |
| EDITIONING_VIEW | VARCHAR2(1) | | Reserved for future use |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| READ_ONLY | VARCHAR2(1) | | Indicates whether the view is read-only (Y) or not (N) |
| CONTAINER_DATA | VARCHAR2(1) | | Indicates whether the view contains container-specific data. Possible values:<br>• Y if the view was created with the CONTAINER_DATA clause<br>• N otherwise |
| BEQUEATH | VARCHAR2(12) | | Possible values:<br>• CURRENT_USER: When the view is a BEQUEATH CURRENT_USER view<br>• DEFINER: When the view is a BEQUEATH DEFINER view<br>For more information about the syntax and semantics of the BEQUEATH clause in the SQL CREATE VIEW statement, see *Oracle Database SQL Language Reference*. |
| ORIGIN_CON_ID | VARCHAR2(256) | | The ID of the container where the data originates. Possible values include:<br>• 0: This value is used for rows in non-CDBs. This value is not used for CDBs.<br>• *n*: This value is used for rows containing data that originate in the container with container ID *n* (*n* = 1 if the row originates in root) |

**See Also:**

- "DBA_VIEWS"
- "USER_VIEWS"

# ALL_VIEWS_AE

ALL_VIEWS_AE describes the views (across all editions) accessible to the current user.

### Related Views

- DBA_VIEWS_AE describes all views (across all editions) in the database.
- USER_VIEWS_AE describes the views (across all editions) owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the view |
| VIEW_NAME | VARCHAR2(128) | NOT NULL | Name of the view |
| TEXT_LENGTH | NUMBER | | Length of the view text |
| TEXT | LONG | | View text. The BEQUEATH clause will not appear as part of the TEXT column in this view. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE_TEXT_LENGTH | NUMBER | | Length of the type clause of the typed view |
| TYPE_TEXT | VARCHAR2(4000) | | Type clause of the typed view |
| OID_TEXT_LENGTH | NUMBER | | Length of the WITH OID clause of the typed view |
| OID_TEXT | VARCHAR2(4000) | | WITH OID clause of the typed view |
| VIEW_TYPE_OWNER | VARCHAR2(128) | | Owner of the type of the view if the view is an typed view |
| VIEW_TYPE | VARCHAR2(128) | | Type of the view if the view is a typed view |
| SUPERVIEW_NAME | VARCHAR2(128) | | Name of the superview, if the view is a subview |
| EDITIONING_VIEW | VARCHAR2(1) | | Indicates whether the view is an editioning view (Y) or not (N) |
| READ_ONLY | VARCHAR2(1) | | Indicates whether the view is read-only (Y) or not (N) |
| EDITION_NAME | VARCHAR2(128) | | Name of the application edition where the object is defined |
| CONTAINER_DATA | VARCHAR2(1) | | Indicates whether the view contains container-specific data. Possible values:<br>• Y if the view was created with the CONTAINER_DATA clause<br>• N otherwise |
| BEQUEATH | VARCHAR2(12) | | Possible values:<br>• CURRENT_USER: When the view is a BEQUEATH CURRENT_USER view<br>• DEFINER: When the view is a BEQUEATH DEFINER view<br>For more information about the syntax and semantics of the BEQUEATH clause in the SQL CREATE VIEW statement, see *Oracle Database SQL Language Reference*. |

**See Also:**

- "DBA_VIEWS_AE"
- "USER_VIEWS_AE"

# ALL_WARNING_SETTINGS

ALL_WARNING_SETTINGS displays information about the warning parameter settings for the objects accessible to the current user.

### Related Views

- DBA_WARNING_SETTINGS displays information about the warning parameter settings for all objects in the database.
- USER_WARNING_SETTINGS displays information about the warning parameter settings for the objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| OBJECT_ID | NUMBER | NOT NULL | Object number of the object |
| OBJECT_TYPE | VARCHAR2(12) | | Type of the object:<br>• PROCEDURE<br>• FUNCTION<br>• PACKAGE<br>• PACKAGE BODY<br>• TRIGGER<br>• TYPE<br>• TYPE BODY |
| WARNING | VARCHAR2(40) | | Warning number or category:<br>• INFORMATIONAL<br>• PERFORMANCE<br>• SEVERE<br>• ALL |
| SETTING | VARCHAR2(7) | | Value of the warning setting:<br>• DISABLE<br>• ENABLE<br>• ERROR |

**See Also:**

- "DBA_WARNING_SETTINGS"
- "USER_WARNING_SETTINGS"

# ALL_XML_INDEXES

ALL_XML_INDEXES describes the XML indexes accessible to the current user.

### Related Views

- DBA_XML_INDEXES describes all XML indexes in the database.
- USER_XML_INDEXES describes the XML indexes owned by the current user. This view does not display the INDEX_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_OWNER | VARCHAR2(128) | NOT NULL | Owner of the XML index |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Name of the XML index |
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the indexed object |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the indexed object |
| TYPE | VARCHAR2(10) | | Type of the indexed column:<br>• REPOSITORY<br>• BINARY<br>• CLOB in OR<br>• CLOB |
| INDEX_TYPE | VARCHAR2(27) | | Type of the index:<br>• STRUCTURED<br>• STRUCTURED and UNSTRUCTURED<br>• UNSTRUCTURED |
| PATH_TABLE_NAME | VARCHAR2(128) | | Name of the path table |
| PARAMETERS | XMLTYPE | | Indexed paths and Scheduler job information |
| ASYNC | VARCHAR2(9) | | Asynchronous index type:<br>• ON-COMMIT<br>• MANUAL<br>• EVERY<br>• ALWAYS |
| STALE | VARCHAR2(5) | | Indicates whether the index type is stale (TRUE) or not (FALSE) |
| PEND_TABLE_NAME | VARCHAR2(128) | | Name of the pending table |
| EX_OR_INCLUDE | VARCHAR2(8) | | Path subsetting:<br>• INCLUDE<br>• EXCLUDE<br>• FULLY IX |

**See Also:**

- "DBA_XML_INDEXES"
- "USER_XML_INDEXES"

# ALL_XML_NESTED_TABLES

ALL_XML_NESTED_TABLES describes all the tables and their corresponding nested tables accessible to the current user.

### Related Views

- DBA_XML_NESTED_TABLES describes all the tables and their corresponding nested tables in the database.

- USER_XML_NESTED_TABLES describes all the tables and their corresponding nested tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| NESTED_TABLE_NAME | VARCHAR2(128) | | Name of the nested table |

**See Also:**

- "DBA_XML_NESTED_TABLES"
- "USER_XML_NESTED_TABLES"

# ALL_XML_OUT_OF_LINE_TABLES

ALL_XML_OUT_OF_LINE_TABLES descibes all the out of line tables connected to a given root table for the same schema accessible to the current user.

### Related Views

- DBA_XML_OUT_OF_LINE_TABLES describes all the out of line tables connected to a given root table for the same schema in the database.
- USER_XML_OUT_OF_LINE_TABLES describes all the out of line tables connected to a given root table for the same schema owned by the current user. This view does not display the TABLE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the out of line table is defined |
| SCHEMA_OWNER | VARCHAR2(128) | | Owner of the schema |
| TABLE_NAME | VARCHAR2(128) | | Name of the out of line table |
| TABLE_OWNER | VARCHAR2(128) | | Owner of the out of line table |

**See Also:**

- "DBA_XML_OUT_OF_LINE_TABLES"
- "USER_XML_OUT_OF_LINE_TABLES"

# ALL_XML_SCHEMA_ATTRIBUTES

ALL_XML_SCHEMA_ATTRIBUTES describes all the attributes and their properties accessible to the current user.

### Related Views

- DBA_XML_SCHEMA_ATTRIBUTES describes all the attributes and their properties accessible to the current user in the database.
- USER_XML_SCHEMA_ATTRIBUTES describes all the attributes and their properties owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | The user who owns the attribute |
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the attribute is defined |
| TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the attribute |
| ATTRIBUTE_NAME | VARCHAR2(2000) | | Name of the attribute |
| IS_REF | NUMBER | | Indicates whether an attribute was defined using a reference in the XML schema (1) or not (0) |
| TYPE_NAME | VARCHAR2(2000) | | Name of the type of the attribute |
| GLOBAL | RAW(1) | | Indicates whether the attribute is global (1) or not (0) |
| ATTRIBUTE | XMLTYPE | | Actual XMLType for the attribute |
| ELEMENT_ID | RAW(20) | | Element ID of the element to which the attribute belongs |
| SQL_TYPE | VARCHAR2(128) | | XDB annotation for sqlType |
| SQL_NAME | VARCHAR2(128) | | XDB annotation value for sqlName |

**See Also:**

- "DBA_XML_SCHEMA_ATTRIBUTES"
- "USER_XML_SCHEMA_ATTRIBUTES"

# ALL_XML_SCHEMA_COMPLEX_TYPES

ALL_XML_SCHEMA_COMPLEX_TYPES describes all complex types accessible to the current user.

### Related Views

- DBA_XML_SCHEMA_COMPLEX_TYPES describes all complex types in the database.
- USER_XML_SCHEMA_COMPLEX_TYPES describes all complex types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | The user who owns the type |
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the type is defined |
| TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the type |
| COMPLEX_TYPE_NAME | VARCHAR2(256) | | Name of the complex type |
| COMPLEX_TYPE | XMLTYPE(XMLSchema "http://xmlns.oracle.com/xdb/XDBSchema.xsd" Element "complexType") | | The actual XMLType of the type |
| BASE_NAME | VARCHAR2(256) | | Name of the base type to which the complex type refers |
| BASE_SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the complex type is defined |
| BASE_TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the type |
| MAINTAIN_DOM | RAW(1) | | XDB annotation for maintainDOM |
| SQL_TYPE | VARCHAR2(128) | | XDB annotation for sqlType |
| SQL_SCHEMA | VARCHAR2(128) | | XDB annotation for sqlSchema |

**See Also:**

- "DBA_XML_SCHEMA_COMPLEX_TYPES"
- "USER_XML_SCHEMA_COMPLEX_TYPES"

# ALL_XML_SCHEMA_ELEMENTS

ALL_XML_SCHEMA_ELEMENTS describes all the elements and their properties accessible to the current user.

### Related Views

- DBA_XML_SCHEMA_ELEMENTS describes all the elements and their properties.
- USER_XML_SCHEMA_ELEMENTS describes all the elements and their properties owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | The user who owns the element |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the element is defined |
| TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the element |
| ELEMENT_NAME | VARCHAR2(2000) | | Name of the element |
| IS_REF | NUMBER | | Indicates whether an attribute was defined using a reference in the XML schema (1) or not (0) |
| TYPE_NAME | VARCHAR2(2000) | | Name of the type of the element |
| GLOBAL | RAW(1) | | Indicates whether the attribute is global (1) or not (0) |
| ELEMENT | XMLTYPE | | The actual XML fragment of the element |
| SQL_INLINE | RAW(1) | | XDB annotation for sqlInline |
| SQL_TYPE | VARCHAR2(128) | | XDB annotation value for sqlType |
| SQL_SCHEMA | VARCHAR2(128) | | XDB annotation value for sqlSchema |
| DEFAULT_TABLE | VARCHAR2(128) | | XDB annotation value for default table |
| SQL_NAME | VARCHAR2(128) | | XDB annotation value for sqlName |
| SQL_COL_TYPE | VARCHAR2(128) | | XDB annotation value for sqlColType |
| MAINTAIN_DOM | RAW(1) | | XDB annotation for maintainDOM |
| MAINTAIN_ORDER | RAW(1) | | XDB annotation for maintainOrder |
| ELEMENT_ID | RAW(20) | | Unique identifier for the element |
| PARENT_ELEMENT_ID | RAW(20) | | Identies the parent of the element |

**See Also:**

- "DBA_XML_SCHEMA_ELEMENTS"
- "USER_XML_SCHEMA_ELEMENTS"

# ALL_XML_SCHEMA_NAMESPACES

ALL_XML_SCHEMA_NAMESPACES describes all the available namespaces accessible to the current user.

### Related Views

- DBA_XML_SCHEMA_NAMESPACES describes all the available namespaces.

- USER_XML_SCHEMA_NAMESPACES describes all the available namespaces owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | User who owns the namespace |
| TARGET_NAMESPACE | VARCHAR2(2000) | | The target namespace |
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema |

**See Also:**

- "DBA_XML_SCHEMA_NAMESPACES"
- "USER_XML_SCHEMA_NAMESPACES"

# ALL_XML_SCHEMA_SIMPLE_TYPES

ALL_XML_SCHEMA_SIMPLE_TYPES describes all simple types accessible to the current user.

### Related Views

- DBA_XML_SCHEMA_SIMPLE_TYPES describes all simple types.
- USER_XML_SCHEMA_SIMPLE_TYPES describes all simple types owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | The user who owns the type |
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the type is defined |
| TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the type |
| SIMPLE_TYPE_NAME | VARCHAR2(256) | | Name of the simple type |
| SIMPLE_TYPE | XMLTYPE(XMLSchema "http://xmlns.oracle.com/xdb/XDBSchema.xsd" Element "simpleType") | | The actual XMLType of the type |

**See Also:**

- "DBA_XML_SCHEMA_SIMPLE_TYPES"
- "USER_XML_SCHEMA_SIMPLE_TYPES"

# ALL_XML_SCHEMA_SUBSTGRP_HEAD

ALL_XML_SCHEMA_SUBSTGRP_HEAD describes the heads of substitution groups accessible to the current user.

### Related Views

- DBA_XML_SCHEMA_SUBSTGRP_HEAD describes the heads of substitution groups.
- USER_XML_SCHEMA_SUBSTGRP_HEAD describes the heads of substitution groups owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | The user who owns the element |
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the element is defined |
| TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the element |
| ELEMENT_NAME | VARCHAR2(256) | | Name of the element |
| ELEMENT | XMLTYPE(XMLSchema "http://xmlns.oracle.com/xdb/XDBSchema.xsd" Element "element") | | The actual XML fragment of the element |

**See Also:**

- "DBA_XML_SCHEMA_SUBSTGRP_HEAD"
- "USER_XML_SCHEMA_SUBSTGRP_HEAD"

# ALL_XML_SCHEMA_SUBSTGRP_MBRS

ALL_XML_SCHEMA_SUBSTGRP_MBRS describes all members of substitution groups accessible to the current user.

### Related Views

- DBA_XML_SCHEMA_SUBSTGRP_MBRS describes all members of substitution groups.
- USER_XML_SCHEMA_SUBSTGRP_MBRS describes all members of substitution groups owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | The user who owns the element |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the element is defined |
| TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the element |
| ELEMENT_NAME | VARCHAR2(256) | | Name of the element |
| ELEMENT | XMLTYPE(XMLSchema "http://xmlns.oracle.com/xdb/XDBSchema.xsd" Element "element") | | The actual XML fragment of the element |
| HEAD_OWNER | VARCHAR2(128) | | The user who owns the head element for the current element |
| HEAD_SCHEMA_URL | VARCHAR2(700) | | The URL of the schema within which the head element exists |
| HEAD_TARGET_NAMESPACE | VARCHAR2(2000) | | The namespace of the head element |
| HEAD_ELEMENT_NAME | VARCHAR2(256) | | Name of the head element |
| HEAD_ELEMENT | XMLTYPE(XMLSchema "http://xmlns.oracle.com/xdb/XDBSchema.xsd" Element "element") | | The actual XMLType of the head element |

**See Also:**

- "DBA_XML_SCHEMA_SUBSTGRP_MBRS"
- "USER_XML_SCHEMA_SUBSTGRP_MBRS"

# ALL_XML_SCHEMAS

ALL_XML_SCHEMAS describes the registered XML schemas accessible to the current user.

### Related Views

- DBA_XML_SCHEMAS describes all registered XML schemas in the database.
- USER_XML_SCHEMAS describes the registered XML schemas owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the XML schema |
| SCHEMA_URL | VARCHAR2(700) | | Schema URL of the XML schema |
| LOCAL | VARCHAR2(3) | | Indicates whether the XML schema is local (YES) or global (NO) |
| SCHEMA | XMLTYPE | | XML schema document |
| INT_OBJNAME | VARCHAR2(4000) | | Internal database object name for the schema |
| QUAL_SCHEMA_URL | VARCHAR2(865) | | Fully qualified schema URL |
| HIER_TYPE | VARCHAR2(11) | | Type of hierarchy for which the schema is enabled:<br>• NONE<br>• RESMETADATA<br>• CONTENTS |
| BINARY | VARCHAR2(3) | | Indicates whether the XML Schema is registered for binary encoding usage (YES) or not (NO) |
| SCHEMA_ID | RAW(16) | | Opaque schema identifier (16 bytes) |
| HIDDEN | VARCHAR2(3) | | Indicates whether the XML Schema has been deleted in hidden mode (YES) or not (NO) |

**See Also:**

- "DBA_XML_SCHEMAS"
- "USER_XML_SCHEMAS"

# ALL_XML_TAB_COLS

ALL_XML_TAB_COLS describes the columns of the XML tables accessible to the current user.

### Related Views

- DBA_XML_TAB_COLS describes the columns of all XML tables in the database.
- USER_XML_TAB_COLS describes the columns of the XML tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the XML table |
| TABLE_NAME | VARCHAR2(128) | | Name of the XML table |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the XML table column |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| XMLSCHEMA | VARCHAR2(700) | | Name of the XML Schema that is used for the table definition |
| SCHEMA_OWNER | VARCHAR2(128) | | Owner of the XML Schema that is used for the table definition |
| ELEMENT_NAME | VARCHAR2(2000) | | Name of the XML SChema element that is used for the table |
| STORAGE_TYPE | VARCHAR2(17) | | Storage option for the XMLType data:<br>• OBJECT-RELATIONAL<br>• BINARY<br>• CLOB<br>**Note:** The CLOB storage option for XMLType data is deprecated in Oracle Database 12*c* Release 1 (12.1). Oracle recommends using the BINARY storage option, instead. |
| ANYSCHEMA | VARCHAR2(3) | | If storage is BINARY, indicates whether the column allows ANYSCHEMA (YES) or not (NO), else NULL |
| NONSCHEMA | VARCHAR2(3) | | If storage is BINARY, indicates whether the column allows NONSCHEMA (YES) or not (NO), else NULL |

**See Also:**

- "DBA_XML_TAB_COLS"
- "USER_XML_TAB_COLS"

# ALL_XML_TABLES

ALL_XML_TABLES describes the XML tables accessible to the current user.

### Related Views

- DBA_XML_TABLES describes all XML tables in the database.
- USER_XML_TABLES describes the XML tables owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the XML table |
| TABLE_NAME | VARCHAR2(128) | | Name of the XML table |
| XMLSCHEMA | VARCHAR2(700) | | Name of the XML Schema that is used for the table definition |
| SCHEMA_OWNER | VARCHAR2(128) | | Owner of the XML Schema that is used for the table definition |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ELEMENT_NAME | VARCHAR2(2000) | | Name of the XML SChema element that is used for the table |
| STORAGE_TYPE | VARCHAR2(17) | | Storage option for the XMLType data:<br>• OBJECT-RELATIONAL<br>• BINARY<br>• CLOB<br>**Note:** The CLOB storage option for XMLType data is deprecated in Oracle Database 12*c* Release 1 (12.1). Oracle recommends using the BINARY storage option, instead. |
| ANYSCHEMA | VARCHAR2(3) | | If storage is BINARY, indicates whether the column allows ANYSCHEMA (YES) or not (NO), else NULL |
| NONSCHEMA | VARCHAR2(3) | | If storage is BINARY, indicates whether the column allows NONSCHEMA (YES) or not (NO), else NULL |

**See Also:**

- "DBA_XML_TABLES"
- "USER_XML_TABLES"

# ALL_XML_VIEW_COLS

ALL_XML_VIEW_COLS describes the columns of the XML views accessible to the current user.

### Related Views

- DBA_XML_VIEW_COLS describes the columns of all XML views in the database.
- USER_XML_VIEW_COLS describes the columns of the XML views owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the XML view |
| VIEW_NAME | VARCHAR2(128) | | Name of the XML view |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the XML view column |
| XMLSCHEMA | VARCHAR2(700) | | Name of the XML Schema that is used for the view definition |
| SCHEMA_OWNER | VARCHAR2(128) | | Owner of the XML Schema that is used for the view definition |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ELEMENT_NAME | VARCHAR2(2000) | | Name of the XML SChema element that is used for the view |

**See Also:**

- "DBA_XML_VIEW_COLS"
- "USER_XML_VIEW_COLS"

# ALL_XML_VIEWS

ALL_XML_VIEWS describes the XML views accessible to the current user.

### Related Views

- DBA_XML_VIEWS describes all XML views the database.
- USER_XML_VIEWS describes the XML views owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the XML view |
| VIEW_NAME | VARCHAR2(128) | | Name of the XML view |
| XMLSCHEMA | VARCHAR2(700) | | Name of the XML Schema that is used for the view definition |
| SCHEMA_OWNER | VARCHAR2(128) | | Owner of the XML Schema that is used for the view definition |
| ELEMENT_NAME | VARCHAR2(2000) | | Name of the XML SChema element that is used for the view |

**See Also:**

- "DBA_XML_VIEWS"
- "USER_XML_VIEWS"

# ALL_XSTREAM_ADMINISTRATOR

ALL_XSTREAM_ADMINISTRATOR displays information about the current users's granted privileges to be an XStream administrator by procedures in the DBMS_XSTREAM_AUTH package.

### Related View

DBA_XSTREAM_ADMINISTRATOR displays information about the users who have been granted privileges to be XStream administrators by procedures in the DBMS_XSTREAM_AUTH package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | | Name of the user who has been granted privileges to be an XStream administrator |
| PRIVILEGE_TYPE | VARCHAR2(7) | | Type of privilege granted:<br>• APPLY<br>• CAPTURE<br>• * - Both APPLY and CAPTURE |
| GRANT_SELECT_PRIVILEGES | VARCHAR2(3) | | Shows whether set of privileges granted to the user (grantee) includes the SELECT_CATALOG_ROLE role, which enables the user to manage other XStream servers that belong to other XStream users.<br>Possible values:<br>• YES: The administrator has the SELECT_CATALOG_ROLE role and other privileges, is considered a full privilege administrator, and can manage other users' XStream configuration<br>• NO: The administrator is considered a minimum privilege administrator, and can only manage XStream configurations where the apply_user or capture_user (based on the PRIVILEGE_TYPE column) matches the username. |
| CREATE_TIME | TIMESTAMP(6) | | Timestamp for the granted privilege |

**See Also:**

- "DBA_XSTREAM_ADMINISTRATOR"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XSTREAM_AUTH package

# ALL_XSTREAM_INBOUND

ALL_XSTREAM_INBOUND displays information about the XStream inbound servers accessible to the current user.

### Related View

DBA_XSTREAM_INBOUND displays information about all XStream inbound servers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SERVER_NAME | VARCHAR2(128) | NOT NULL | Name of the inbound server |
| QUEUE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue associated with the inbound server |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue associated with the inbound server |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| APPLY_USER | VARCHAR2(128) | | Name of the user who can connect to the inbound server and apply messages |
| USER_COMMENT | VARCHAR2(4000) | | User comment |
| CREATE_DATE | TIMESTAMP(6) | | Date when the inbound server was created |
| STATUS | VARCHAR2(8) | | Status of the inbound server:<br>• DISABLED - The inbound server is not running.<br>• DETACHED - The inbound server is running, but the XStream client application is not attached to it.<br>• ATTACHED - The inbound server is running, and the XStream client application is attached to it.<br>• ABORTED - The inbound server became disabled because it encountered an error. |
| COMMITTED_DATA_ONLY | VARCHAR2(3) | | YES - means the inbound server can receive only LCRs in committed transactions from the XStream client application. A committed transaction is an assembled, noninterleaving transaction with no rollbacks. |

**See Also:**

"DBA_XSTREAM_INBOUND"

# ALL_XSTREAM_INBOUND_PROGRESS

ALL_XSTREAM_INBOUND_PROGRESS displays information about the progress made by the XStream inbound servers accessible to the current user.

### Related View

DBA_XSTREAM_INBOUND_PROGRESS displays information about the progress made by all XStream inbound servers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SERVER_NAME | VARCHAR2(128) | NOT NULL | Name of the inbound server |
| PROCESSED_LOW_POSITION | RAW(64) | | Position of the processed low transaction |
| APPLIED_LOW_POSITION | RAW(64) | | All messages with a commit position less than this value have been applied |
| APPLIED_HIGH_POSITION | RAW(64) | | Highest commit position of a transaction that has been applied |
| SPILL_POSITION | RAW(64) | | Position of the spill low watermark of the transactions currently being applied |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OLDEST_POSITION | RAW(64) | | Earliest position of the transactions currently being applied |
| OLDEST_MESSAGE_NUMBER | NUMBER | NOT NULL | Earliest message number of the transactions currently being applied |
| APPLIED_MESSAGE_NUMBER | NUMBER | NOT NULL | Message number up to which all transactions have definitely been applied. This value is the low watermark for the inbound server. That is, messages with a commit message number less than or equal to this message number have definitely been applied, but some messages with a higher commit message number may also have been applied. |
| APPLIED_TIME | DATE | | Time at which the message with the message number displayed in the APPLIED_MESSAGE_NUMBER column was applied |
| APPLIED_MESSAGE_CREATE_TIME | DATE | | Time at which the message with the message number displayed in the APPLIED_MESSAGE_NUMBER column was created at its source database |
| SPILL_MESSAGE_NUMBER | NUMBER | | Spill low watermark. Any message with a lower SCN has either been applied or spilled to disk. The XStream client application does not need to send logical change records (LCRs) with a lower SCN than the spill low watermark. Spilled messages may not have been applied yet. |
| SOURCE_DATABASE | VARCHAR2(128) | NOT NULL | Database where the transaction originated |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |

**See Also:**

"DBA_XSTREAM_INBOUND_PROGRESS"

# ALL_XSTREAM_OUT_SUPPORT_MODE

ALL_XSTREAM_OUT_SUPPORT_MODE displays information about the level of XStream capture process support for the tables accessible to the current user in the database.

### Related View

DBA_XSTREAM_OUT_SUPPORT_MODE displays information about the level of XStream capture process support for the tables in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Table owner |
| OBJECT_NAME | VARCHAR2(128) | | Table name |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SUPPORT_MODE | VARCHAR2(6) | | Capture process support level for the table:<br>• FULL - A capture process can capture changes made to all of the columns in the table.<br>• ID KEY - A capture process can capture changes made to the key columns and any other columns in the table that are supported by the capture process, except for LOB, LONG, LONG RAW, and XMLType columns.<br>• NONE - A capture process cannot capture changes made to any columns in the table. |

**See Also:**

"DBA_XSTREAM_OUT_SUPPORT_MODE"

# ALL_XSTREAM_OUTBOUND

ALL_XSTREAM_OUTBOUND displays information about the XStream outbound servers accessible to the current user.

### Related View

DBA_XSTREAM_OUTBOUND displays information about all XStream outbound servers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SERVER_NAME | VARCHAR2(128) | NOT NULL | Name of the outbound server |
| CONNECT_USER | VARCHAR2(128) | | Name of the user who can connect to the outbound server and process the outbound LCRs |
| CAPTURE_NAME | VARCHAR2(128) | | Name of the Streams capture process |
| SOURCE_DATABASE | VARCHAR2(128) | | Database where the transaction originated |
| CAPTURE_USER | VARCHAR2(128) | | Current user who is enqueuing captured messages |
| QUEUE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the queue associated with the outbound server |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue associated with the outbound server |
| USER_COMMENT | VARCHAR2(4000) | | User comment |
| CREATE_DATE | TIMESTAMP(6) | | Date when the outbound server was created |
| STATUS | VARCHAR2(8) | | Status of the outbound server:<br>• DISABLED - The outbound server is not running.<br>• DETACHED - The outbound server is running, but the XStream client application is not attached to it. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • ATTACHED - The outbound server is running, and the XStream client application is attached to it.<br>• ABORTED - The outbound server became disabled because it encountered an error. |
| COMMITTED_DATA_ONLY | VARCHAR2(3) | | YES if the outbound server can send only LCRs in committed transactions to the XStream client application. A committed transaction is an assembled, noninterleaving transaction with no rollbacks.<br>NO if the outbound server can send LCRs in transactions that have not yet committed to the XStream client application. This mode is for internal Oracle use only. |
| START_SCN | NUMBER | | The SCN from which the outbound server's capture process started capturing changes when it was last started |
| START_TIME | TIMESTAMP(6) | | The time from which the outbound server's capture process started capturing changes when it was last started |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |

**See Also:**

"DBA_XSTREAM_OUTBOUND"

# ALL_XSTREAM_OUTBOUND_PROGRESS

ALL_XSTREAM_OUTBOUND_PROGRESS displays information about the progress made by the XStream outbound servers accessible to the current user.

### Related View

DBA_XSTREAM_OUTBOUND_PROGRESS displays information about the progress made by all XStream outbound servers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SERVER_NAME | VARCHAR2(128) | NOT NULL | Name of the outbound server |
| SOURCE_DATABASE | VARCHAR2(128) | | Global name of the database where the transaction originated. For a PDB, this is the global name of the PDB. |
| PROCESSED_LOW_POSITION | RAW(64) | | Position of the low-watermark transaction processed by the outbound server |
| PROCESSED_LOW_TIME | DATE | | Time when the processed low position was last updated by the outbound server |
| OLDEST_POSITION | RAW(64) | | The position of the earliest LCR that is required by the XStream client application |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |
| PROCESSED_LOW_SCN | NUMBER | NOT NULL | SCN of the processed low transaction |
| OLDEST_SCN | NUMBER | NOT NULL | Oldest SCN of the transactions currently being captured |

**See Also:**

"DBA_XSTREAM_OUTBOUND_PROGRESS"

# ALL_XSTREAM_RULES

ALL_XSTREAM_RULES displays information about the XStream rules accessible to the current user.

### Related View

DBA_XSTREAM_RULES displays information about all XStream server rules in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAMS_NAME | VARCHAR2(128) | | Name of the XStream process |
| STREAMS_TYPE | VARCHAR2(12) | | Type of the XStream process:<br>• CAPTURE<br>• APPLY |
| STREAMS_RULE_TYPE | VARCHAR2(9) | | The XStream type of the rule:<br>• TABLE<br>• SCHEMA<br>• GLOBAL |
| RULE_SET_OWNER | VARCHAR2(128) | | Owner of the rule set |
| RULE_SET_NAME | VARCHAR2(128) | | Name of the rule set |
| RULE_SET_TYPE | CHAR(8) | | Type of the rule set:<br>• POSITIVE<br>• NEGATIVE |
| RULE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | NOT NULL | Name of the rule |
| RULE_TYPE | VARCHAR2(9) | | The type of the rule:<br>• DML<br>• DDL |
| RULE_CONDITION | CLOB | | Current rule condition |
| SCHEMA_NAME | VARCHAR2(128) | | For table and schema rules, the schema name |
| OBJECT_NAME | VARCHAR2(128) | | For table rules, the table name |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INCLUDE_TAGGED_LCR | VARCHAR2(3) | | Indicates whether to include tagged LCRs (YES) or not (NO) |
| SUBSETTING_OPERATION | VARCHAR2(6) | | For subset rules, the type of operation:<br>• INSERT<br>• UPDATE<br>• DELETE |
| DML_CONDITION | VARCHAR2(4000) | | For subset rules, the row subsetting condition |
| SOURCE_DATABASE | VARCHAR2(128) | | The global name of the database where the LCRs originated. In a PDB, this is the global name of the PDB. |
| ORIGINAL_RULE_CONDITION | VARCHAR2(4000) | | For rules created by the XStream administrative APIs, the original rule condition when the rule was created |
| SAME_RULE_CONDITION | VARCHAR2(3) | | For rules created by the XStream administrative APIs, indicates whether the current rule condition is the same as the original rule condition (YES) or not (NO) |
| SOURCE_ROOT_NAME | VARCHAR2(128) | | The global name of the source root database |
| SOURCE_CONTAINER_NAME | VARCHAR2(128) | | The container name of the database where the transactions originated |

**See Also:**

"DBA_XSTREAM_RULES"

# ALL_XSTREAM_TRANSFORMATIONS

ALL_XSTREAM_TRANSFORMATIONS displays information about all XStream transformations accessible to the current user, in order of execution.

### Related View

DBA_XSTREAM_TRANSFORMATIONS displays information about all XStream transformations available on a system, in order of execution.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule which has an associated transformation |
| RULE_NAME | VARCHAR2(128) | | Name of the rule which has an associated transformation |
| TRANSFORM_TYPE | VARCHAR2(26) | | Type of the transformation:<br>• DECLARATIVE TRANSFORMATION<br>• SUBSET RULE<br>• CUSTOM TRANSFORMATION |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FROM_SCHEMA_NAME | VARCHAR2(128) | | Schema to be renamed |
| TO_SCHEMA_NAME | VARCHAR2(128) | | New schema name |
| FROM_TABLE_NAME | VARCHAR2(128) | | Table to be renamed |
| TO_TABLE_NAME | VARCHAR2(128) | | New table name |
| SCHEMA_NAME | VARCHAR2(128) | | Schema of the column to be modified |
| TABLE_NAME | VARCHAR2(128) | | Table of the column to be modified |
| FROM_COLUMN_NAME | VARCHAR2(4000) | | Column to be renamed |
| TO_COLUMN_NAME | VARCHAR2(4000) | | New column name |
| COLUMN_NAME | VARCHAR2(4000) | | Column to add or delete |
| COLUMN_VALUE | ANYDATA | | Value of the column to add |
| COLUMN_TYPE | VARCHAR2(4000) | | Type of the new column |
| COLUMN_FUNCTION | VARCHAR2(128) | | Name of the default function used to add a column |
| VALUE_TYPE | VARCHAR2(3) | | Indicates whether to modify the old (OLD), new (NEW), or both (*) values of the LCR |
| USER_FUNCTION_NAME | VARCHAR2(4000) | | Name of the user-defined transformation function to run |
| SUBSETTING_OPERATION | VARCHAR2(6) | | DML operation for row subsetting:<br>• INSERT<br>• UPDATE<br>• DELETE |
| DML_CONDITION | VARCHAR2(4000) | | Row subsetting condition |
| DECLARATIVE_TYPE | VARCHAR2(13) | | Type of declarative transform to run:<br>• KEEP COLUMNS<br>• DELETE COLUMN<br>• RENAME COLUMN<br>• ADD COLUMN<br>• RENAME TABLE<br>• RENAME SCHEMA |
| PRECEDENCE | NUMBER | | Execution order relative to other declarative transformations on the same STEP_NUMBER |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

**See Also:**

"DBA_XSTREAM_TRANSFORMATIONS"

# ALL_ZONEMAP_MEASURES

ALL_ZONEMAP_MEASURES describes the measures for all zone maps accessible to the user.

### Related Views

- DBA_ZONEMAP_MEASURES describes the measures for all the zone maps in the database.
- USER_ZONEMAP_MEASURES describes the measures for all the zone maps owned by the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the zone map |
| ZONEMAP_NAME | VARCHAR2(128) | NOT NULL | Name of the zone map |
| MEASURE | LONG | | Column whose MIN/MAX value is computed |
| POSITION_IN_SELECT | NUMBER | NOT NULL | Original position of the mesaure aggregate on the SELECT list of zone map defining query |
| AGG_FUNCTION | VARCHAR2(13) | | Name of aggregate in zone map table |
| AGG_COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column whose MIN/MAX per zone maintained |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2), and is intended for use with Oracle Exadata release 12.1.2.1.1 or later.

**See Also:**

- "DBA_ZONEMAP_MEASURES"
- "USER_ZONEMAP_MEASURES"
- *Oracle Database Data Warehousing Guide* for more information about zone maps

# ALL_ZONEMAPS

ALL_ZONEMAPS describes all the zone maps accessible to the user.

### Related Views

- DBA_ZONEMAPS describes all the zone maps in the database.
- USER_ZONEMAPS describes all the zone maps owned by the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the zone map |
| ZONEMAP_NAME | VARCHAR2(128) | | Name of the zone map |
| FACT_OWNER | VARCHAR2(128) | | Owner of the fact table of the zone map |
| FACT_TABLE | VARCHAR2(128) | | Name of the fact table on which the zone map is defined |
| SCALE | NUMBER | | Scale factor of the zone map |
| HIERARCHICAL | VARCHAR2(12) | | Indicates whether the zone map is hierarchical (YES) or not (NO) |
| WITH_CLUSTERING | VARCHAR2(15) | | Indicates whether the zone map is created with the CLUSTERING clause (YES) or not (NO) |
| QUERY | LONG | | Zone map defining query |
| QUERY_LEN | NUMBER(38) | | Length of defining query in bytes |
| PRUNING | VARCHAR2(8) | | Indicates whether the zone map is enabled for pruning (ENABLED) or not (DISABLED) |
| REFRESH_MODE | VARCHAR2(17) | | Refresh mode for the zone map:<br>• COMMIT<br>• DEMAND<br>• DATAMOVEMENT<br>• LOAD<br>• LOAD DATAMOVEMENT |
| REFRESH_METHOD | VARCHAR2(14) | | Refresh method for the zone map<br>• COMPLETE<br>• FORCE<br>• FAST |
| LAST_REFRESH_METHOD | VARCHAR2(19) | | The last refresh method used for the zone map:<br>• NA<br>• COMPLETE<br>• FAST<br>• ERROR-UNKNOWN |
| LAST_REFRESH_TIME | TIMESTAMP(9) | | Time of the last refresh |
| INVALID | VARCHAR2(7) | | Indicates whether the zone map is invalid due to some DDL (YES) or not (NO) |
| STALE | VARCHAR2(7) | | Indicates whether the zone map is stale because of DML operations and cannot be used for pruning (YES) or not (NO) or whether this cannot be determined (UNKNOWN) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| UNUSABLE | VARCHAR2(8) | | Indicates whether the zone map has been marked unusable by the owner (YES) or not (NO) |
| COMPILE_STATE | VARCHAR2(19) | | Current compile state of the zone map:<br>• VALID<br>• AUTHORIZATION_ERROR<br>• COMPILATION_ERROR<br>• NEEDS_COMPILE<br>• ERROR_UNKNOWN<br>Similar to ALL_MVIEWS.COMPILE_STATE. |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2), and is intended for use with Oracle Exadata release 12.1.2.1.1 or later.

**See Also:**

- "DBA_ZONEMAPS"
- "USER_ZONEMAPS"
- *Oracle Database Data Warehousing Guide* for more information about zone maps

# AUDIT_ACTIONS

AUDIT_ACTIONS describes audit trail action type codes. This table can be used to map action type numbers to action type names.

**Note:**

The mapping explained in this view is valid for audit trail records from the following views only, and such audit records are generated only when unified auditing is not enabled:

- DBA_AUDIT_TRAIL
- DBA_COMMON_AUDIT_TRAIL
- DBA_FGA_AUDIT_TRAIL
- USER_AUDIT_TRAIL
- V$XML_AUDIT_TRAIL

**See Also:**

- *Oracle Database Security Guide* for more information about unified auditing.
- *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ACTION | NUMBER | NOT NULL | Numeric audit trail action type code. |
| NAME | VARCHAR2(28) | NOT NULL | Name of the type of audit trail action |

# AUDIT_UNIFIED_CONTEXTS

AUDIT_UNIFIED_CONTEXTS describes the application context's attributes, which are configured to be captured in the audit trail.

**Note:**

This view is populated only in an Oracle Database where unified auditing is enabled.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAMESPACE | VARCHAR2(128) | | Application context namespace |
| ATTRIBUTE | VARCHAR2(128) | | Application context attribute |
| USER_NAME | VARCHAR2(128) | | Username of database user for whom the application context's attribute is confiured to be captured in the audit trail |

# AUDIT_UNIFIED_ENABLED_POLICIES

AUDIT_UNIFIED_ENABLED_POLICIES describes all the audit policies that are enabled in the database.

**Note:**

This view is populated only in an Oracle Database where unified auditing is enabled.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_NAME | VARCHAR2(128) | | Username of the database user for whom the audit policy is enabled |
| POLICY_NAME | VARCHAR2(128) | | Name of the audit policy |
| ENABLED_OPT | VARCHAR2(8) | | Enabled option of the audit policy. The possible values are BY, EXCEPT, and DISABLED. |
| SUCCESS | VARCHAR2(3) | | Indicates whether the audit policy is enabled for auditing successful events (YES) or not (NO) |
| FAILURE | VARCHAR2(3) | | Indicates whether the audit policy is enabled for auditing unsuccessful events (YES) or not (NO) |

# AUDIT_UNIFIED_POLICIES

AUDIT_UNIFIED_POLICIES describes all audit policies created in the database.

**Note:**

This view is populated only in an Oracle Database where unified auditing is enabled.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| POLICY_NAME | VARCHAR2(128) | | Name of the audit policy |
| AUDIT_CONDITION | VARCHAR2(4000) | | Condition associated with the audit policy |
| CONDITION_EVAL_OPT | VARCHAR2(9) | | Evaluation option associated with the audit policy's condition. The possible values are STATEMENT, SESSION, INSTANCE, and NONE. |
| AUDIT_OPTION | VARCHAR2(128) | | Auditing option defined in the audit policy |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AUDIT_OPTION_TYPE | VARCHAR2(16) | | Type of the auditing option. The possible values are SYSTEM PRIVILEGE, SYSTEM ACTION, OBJECT ACTION, ROLE PRIVILEGE, and INVALID. |
| OBJECT_SCHEMA | VARCHAR2(128) | | Owner of the object, for an object-specific auditing option |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object, for an object-specific auditing option |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object, for an object-specific auditing option |
| COMMON | VARCHAR2(3) | | Indicates whether the audit policy is a common audit policy or local audit policy. The value is NULL for a non-CDB. |

# AUDIT_UNIFIED_POLICY_COMMENTS

AUDIT_UNIFIED_POLICY_COMMENTS shows the description of each unified audit policy, if a description was entered for the unified audit policy using the COMMENT SQL statement.

**Note:**

This view is populated only in an Oracle Database where unified auditing is enabled.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| POLICY_NAME | VARCHAR2(128) | NOT NULL | Name of the unified audit policy |
| COMMENTS | VARCHAR2(4000) | | Description of the unified audit policy, if one was entered using the COMMENT SQL statement |

# AUDITABLE_SYSTEM_ACTIONS

AUDITABLE_SYSTEM_ACTIONS maps the auditable system action numbers to the action names. These actions are configurable for audit when unified auditing is enabled.

The actions include:

- All standard RDBMS actions (from the V$SQLCOMMAND view) except the following, which are not configurable for auditing:
  - ALTER EDITION

  - ALTER REWRITE EQUIVALENCE
  - CREATE BITMAPFILE
  - CREATE CONTROL FILE
  - CREATE DATABASE
  - DECLARE REWRITE EQUIVALENCE
  - DROP BITMAPFILE
  - DROP DATABASE
  - DROP REWRITE EQUIVALENCE
  - FLASHBACK DATABASE
  - NO-OP (No operation)
- Other actions:
  - ALL
  - LOGON
  - LOGOFF

**Note:**

This view is populated only in an Oracle Database where unified auditing is enabled.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE | NUMBER | | Numeric component type for system wide actions |
| COMPONENT | VARCHAR2(64) | | Name of component for system wide actions |
| ACTION | NUMBER | | Numeric auditable action code for system wide actions |
| NAME | VARCHAR2(64) | | Name of auditable action |

**See Also:**

"V$SQLCOMMAND"

# CAT

CAT is a synonym for USER_CATALOG.

**See Also:**

"USER_CATALOG"

# CATALOG

CATALOG is included for compatibility. Oracle recommends that you not use this view.

# CHAINED_ROWS

CHAINED_ROWS stores the output for the ANALYZE statement with the LIST CHAINED ROWS clause. You must run the utlchain.sql or utlchn1.sql script to create this table.

| Column | Description |
|---|---|
| OWNER_NAME | Table owner |
| TABLE_NAME | Table name |
| CLUSTER_NAME | Cluster the table is in, if any |
| PARTITION_NAME | The name of the partition |
| SUBPARTITION_NAME | The name of the subpartition |
| HEAD_ROWID | ROWID the chained row is accessed by |
| ANALYZE_TIMESTAMP | Date/time that the ANALYZE statement was issued |

# CLIENT_RESULT_CACHE_STATS$

CLIENT_RESULT_CACHE_STATS$ displays various Client Result Cache settings and usage statistics. Statistics are stored as name, value pairs. For each client cache ID, there will be multiple rows for each statistic.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CACHE_ID | NUMBER | NOT NULL | Unique ID per client cache |
| STAT_ID | NUMBER | NOT NULL | Statistic ID |
| NAME | VARCHAR2(128) | | Name of the statistic (see Table 3-1) |
| VALUE | NUMBER | | Value of the statistic |

***Table 3-1 CLIENT_RESULT_CACHE_STATS$ Statistics***

| Statistic Name | Description |
|---|---|
| Block Size | Size (in bytes) of each memory block in the result cache. |

| Statistic Name | Description |
|---|---|
| Block Count Max | Maximum number of blocks that can be allocated in the result cache based on the cache size configuration parameters on server and on client. |
| Block Count Current | Current number of blocks allocated by the client result cache. |
| Hash Bucket Count | Size of the hash table used for query matching. |
| Create Count Success | Number of cached result sets that did not get invalidated before caching all the rows of the result set. |
| Create Count Failure | Number of cached result sets that did not fetch all the rows in the result set. |
| Find Count | Number of cache hits. |
| Invalidation Count | Number of cached result sets that got invalidated due to database changes that could have affected the result set. |
| Delete Count Invalid | Number of cached result rests not invalidated whose memory was reclaimed by result cache. |
| Delete Count Valid | Number of invalidated cached result rests whose memory was reclaimed by result cache. |

# CLU

CLU is a synonym for USER_CLUSTERS.

**See Also:**

"USER_CLUSTERS"

# COL

COL is included for compatibility. Oracle recommends that you not use this view.

# COLS

COLS is a synonym for USER_TAB_COLUMNS.

**See Also:**

"USER_TAB_COLUMNS"

# DATABASE_EXPORT_OBJECTS

DATABASE_EXPORT_OBJECTS lists simple path names for some of the object types belonging to a full Data Pump export, which is invoked using the FULL=Y parameter on the expdp command. Users of the Data Pump Export and Import utilities can query this view to determine valid values for the EXCLUDE and INCLUDE parameters.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_PATH | VARCHAR2(200) | NOT NULL | Simple path name for the object type |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMMENTS | VARCHAR2(2000) | | Comment on the object type |
| NAMED | CHAR(1) | | Do objects of this type have names? If yes (Y), then the name can be specified in the optional name_clause on the EXCLUDE and INCLUDE parameters. |

**See Also:**

- "SCHEMA_EXPORT_OBJECTS"
- "TABLE_EXPORT_OBJECTS"
- *Oracle Database Utilities* for more information on performing a full Data Pump export using the `expdp` command

# DATABASE_PROPERTIES

DATABASE_PROPERTIES lists permanent database properties.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROPERTY_NAME | VARCHAR2(128) | NOT NULL | Property name |
| PROPERTY_VALUE | VARCHAR2(4000) | | Property value |
| DESCRIPTION | VARCHAR2(4000) | | Property description |

**Note:**

The CDB_PROPERTIES view provides access to data visible to PDBs through the DATABASE_PROPERTIES view.

# 4

# Static Data Dictionary Views: DBA_2PC_NEIGHBORS to DBA_HIST_JAVA_POOL_ADVICE

This chapter contains the static data dictionary views DBA_2PC_NEIGHBORS through DBA_HIST_JAVA_POOL_ADVICE.

## DBA_2PC_NEIGHBORS

DBA_2PC_NEIGHBORS describes incoming and outgoing connections for pending transactions.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOCAL_TRAN_ID | VARCHAR2(22) | | Local identifier of a transaction |
| IN_OUT | VARCHAR2(3) | | IN for incoming connections, OUT for outgoing |
| DATABASE | VARCHAR2(128) | | IN for client database name, OUT for outgoing database link |
| DBUSER_OWNER | VARCHAR2(128) | | IN for name of local user, OUT for owner of database link |
| INTERFACE | VARCHAR2(1) | | C for request commit, otherwise N for prepare or request read only commit |
| DBID | VARCHAR2(16) | | Database ID at the other end of the connection |
| SESS# | NUMBER(38) | | Session number of the connection at this database |
| BRANCH | VARCHAR2(128) | | Transaction branch ID of the connection at this database |

## DBA_2PC_PENDING

DBA_2PC_PENDING describes distributed transactions awaiting recovery.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOCAL_TRAN_ID | VARCHAR2(22) | NOT NULL | String of form: n.n.n; n is a number |
| GLOBAL_TRAN_ID | VARCHAR2(169) | | Globally unique transaction ID |
| STATE | VARCHAR2(16) | NOT NULL | Collecting, prepared, committed, forced commit, or forced rollback |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MIXED | VARCHAR2(3) | | YES indicates part of the transaction committed and part rolled back |
| ADVICE | VARCHAR2(1) | | C for commit, R for rollback, else NULL |
| TRAN_COMMENT | VARCHAR2(255) | | Text for commit work comment text |
| FAIL_TIME | DATE | NOT NULL | Value of SYSDATE when the row was inserted (transaction or system recovery) |
| FORCE_TIME | DATE | | Time of manual force decision (null if not forced locally) |
| RETRY_TIME | DATE | NOT NULL | Time automatic recovery (RECO) last tried to recover the transaction |
| OS_USER | VARCHAR2(64) | | Operating system-specific name for the end-user |
| OS_TERMINAL | VARCHAR2(255) | | Operating system-specific name for the end-user terminal |
| HOST | VARCHAR2(128) | | Name of the host system for the end-user |
| DB_USER | VARCHAR2(128) | | Oracle user name of the end-user at the topmost database |
| COMMIT# | VARCHAR2(16) | | Global commit number for committed transactions |

# DBA_ACL_NAME_MAP

DBA_ACL_NAME_MAP maps new names of the access control lists for PL/SQL network utility packages from old XDB names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| XDB_NAME | VARCHAR2(4000) | NOT NULL | The old XDB name of the access control list |
| ACL | VARCHAR2(128) | | The new name of the access control list |
| ACL_OWNER | VARCHAR2(128) | | The owner of the access control list |

# DBA_ADDM_FDG_BREAKDOWN

DBA_ADDM_FDG_BREAKDOWN describes the contribution for each finding from the different instances. There is one row for each finding and for each instance participating in the analysis. Rows are omitted if the impact from that instance is not sufficient to register a finding in a local ADDM analysis.

### Related View

USER_ADDM_FDG_BREAKDOWN describes the contribution for each finding from the different instances owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER | NOT NULL | Identifies the task to which this finding belongs (see DBA_ADVISOR_TASKS) |
| FINDING_ID | NUMBER | NOT NULL | Identifies the finding (see DBA_ADVISOR_FINDINGS) |
| INSTANCE_NUMBER | NUMBER | NOT NULL | The number of the instance contributing to the finding |
| DATABASE_TIME | NUMBER | | The database time, in microseconds, accumulated by this instance during the analysis period |
| ACTIVE_SESSIONS | NUMBER | | The average number of active sessions of the finding in this instance |
| PERC_ACTIVE_SESSIONS | NUMBER | | The percentage of contribution from this instance compared to the total impact of the finding |

**See Also:**

"USER_ADDM_FDG_BREAKDOWN"

# DBA_ADDM_FINDINGS

DBA_ADDM_FINDINGS displays the ADDM findings discovered by all advisors in the database. Each row for ADDM tasks in the related DBA_ADVISOR_FINDINGS view has a corresponding row in this view.

### Related View

USER_ADDM_FINDINGS displays the ADDM findings discovered by the advisors owned by the current user. Each row for ADDM tasks in the related USER_ADVISOR_FINDINGS view has a corresponding row in this view. The USER_ADDM_FINDINGS view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |
| FINDING_ID | NUMBER | NOT NULL | Identifier of the finding |
| FINDING_NAME | VARCHAR2(4000) | | Name of the finding |
| TYPE | VARCHAR2(11) | | Type of the finding:<br>PROBLEM<br>SYMPTOM<br>ERROR |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | INFORMATION |
| TYPE_ID | NUMBER | NOT NULL | Numeric ID for the value in column TYPE |
| PARENT | NUMBER | NOT NULL | Identifier of the parent finding |
| OBJECT_ID | NUMBER | | Identifier of the associated object, if any |
| IMPACT_TYPE | VARCHAR2(4000) | | Impact of the finding on the system |
| IMPACT | NUMBER | | Impact value |
| MESSAGE | VARCHAR2(4000) | | Message describing the finding |
| MORE_INFO | VARCHAR2(4000) | | Additional info associated with the finding |
| FILTERED | VARCHAR2(1) | | A value of Y means that the row in the view was filtered out by a directive (or a combination of directives). A value of N means that the row was not filtered. |
| FLAGS | NUMBER | | For internal use only by advisor framework clients |
| DATABASE_TIME | NUMBER | | The database time, in microseconds, accumulated by this finding |
| ACTIVE_SESSIONS | NUMBER | | The average number of active sessions for the finding |
| PERC_ACTIVE_SESS | NUMBER | | The percentage of active sessions for this finding out of the total active sessions for the task |
| IS_AGGREGATE | CHAR(1) | | A value of Y means that this finding is created for global/continental ADDM as an aggregate of local ADDM findings. Otherwise, the value is N. |
| METER_LEVEL | VARCHAR2(6) | | Reserved for future use |
| QUERY_IS_APPROX | CHAR(1) | | Indicates whether the ASH SQL associated with the finding is an approximate query (Y) or an exact query (N).<br>To get the associated query, use the TASK_NAME and FINDING_ID columns from this view and call the PL/SQL function DBMS_ADDM.GET_ASH_QUERY(task_name, finding_id). |

**See Also:**

- "USER_ADDM_FINDINGS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ADDM.GET_ASH_QUERY procedure

# DBA_ADDM_INSTANCES

DBA_ADDM_INSTANCES displays instance-level information for ADDM tasks that finished executing. For each instance that was supposed to be analyzed (whether it was or not) there is one row describing information about it.

### Related View

USER_ADDM_INSTANCES displays instance-level information for ADDM tasks that finished executing in all instances owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER | NOT NULL | The ID of the main ADDM task |
| INSTANCE_NUMBER | NUMBER | NOT NULL | The number of the instance |
| INSTANCE_NAME | VARCHAR2(16) | | The name of the instance |
| HOST_NAME | VARCHAR2(64) | | The name of the system on which the instance was running |
| STATUS | VARCHAR2(10) | | How information from this instance was used by the ADDM task.<br>A value of ANALYZED means that the instance participated fully in the analysis. For the following remaining values, the instance was not used during task execution, for the stated reason:<br>BOUNCED - the instance was shut down or started during the analysis period<br>NO_SNAPS - there were either begin or end snapshots missing for the instance<br>NO_STATS - there were key statistics missing for the instance<br>NOT_FOUND - no mention of this instance could be found in AWR during the analysis period |
| DATABASE_TIME | NUMBER | | The database time, in microseconds, accumulated by this instance during the analysis period |
| ACTIVE_SESSIONS | NUMBER | | The average number of active sessions for the instance during the analysis period |
| PERC_ACTIVE_SESS | NUMBER | | The percentage of active sessions for this instance, out of the total active sessions for the task |
| METER_LEVEL | VARCHAR2(6) | | Reserved for future use |
| LOCAL_TASK_ID | NUMBER | | The ID of a local ADDM task that contained an analysis of the instance for the same analysis period as that of the main task. If the main task is a local ADDM, then this value is the same as the TASK_ID value. |

**See Also:**

"USER_ADDM_INSTANCES"

# DBA_ADDM_SYSTEM_DIRECTIVES

02/03/09. Added per bug 6374236; info from Uri Shaft.

DBA_ADDM_SYSTEM_DIRECTIVES displays information about global instances for ADDM system directives.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INSTANCE_ID | NUMBER | NOT NULL | Unique ID for the directive instance. The directive management engine automatically generates ID numbers. |
| INSTANCE_NAME | VARCHAR2(128) | NOT NULL | User-assigned name for the directive instance. |
| DIRECTIVE_NAME | VARCHAR2(128) | NOT NULL | Any value that further classifies this directive within a domain. The domain and the name form a unique key for the directive. |
| DESCRIPTION | VARCHAR2(4000) | | Description of the ADDM system directive, shown in the language used by the current session |

# DBA_ADDM_TASK_DIRECTIVES

DBA_ADDM_TASK_DIRECTIVES displays information about all ADDM task directives in the database.

### Related View

USER_ADDM_TASK_DIRECTIVES displays information about ADDM task directives owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER | | An ADDM advisor task identifier to which the directive instance is associated |
| TASK_NAME | VARCHAR2(128) | | An ADDM advisor task to which the directive instance is associated |
| USERNAME | VARCHAR2(128) | NOT NULL | Database user who owns the ADDM task instance |
| SEQ_ID | NUMBER | NOT NULL | Unique ID for the directive instance. The directive management engine automatically generates ID numbers. |
| INSTANCE_NAME | VARCHAR2(128) | NOT NULL | A user-assigned name for the ADDM task directive instance |
| DIRECTIVE_NAME | VARCHAR2(128) | NOT NULL | Any value that further classifies this directive within a domain. The domain and the name form a unique key for the directive. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DESCRIPTION | VARCHAR2(4000) | | Description of the ADDM task directive, shown in the language used by the current session |

**See Also:**

"USER_ADDM_TASK_DIRECTIVES"

# DBA_ADDM_TASKS

DBA_ADDM_TASKS displays information about all ADDM tasks in the database. The view contains one row for each row in the related DBA_ADVISOR_TASKS view that has ADVISOR_NAME=ADDM and STATUS=COMPLETED.

### Related View

USER_ADDM_TASKS displays information about the ADDM tasks owned by the current user. The view contains one row for each row in the related USER_ADVISOR_TASKS view that has ADVISOR_NAME=ADDM and STATUS=COMPLETED. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| DESCRIPTION | VARCHAR2(256) | | User-supplied description of the task |
| ADVISOR_NAME | VARCHAR2(128) | | Advisor associated with the task |
| CREATED | DATE | NOT NULL | Creation date of the task |
| LAST_MODIFIED | DATE | NOT NULL | Date on which the task was last modified |
| PARENT_TASK_ID | NUMBER | | Identifier of the parent task (if the task was created because of the recommendation of another task) |
| PARENT_RXEC_ID | NUMBER | | Identifier of the recommendation within the parent task that resulted in the creation of the task |
| LAST_EXECUTION | VARCHAR2(128) | | Name of the current or last task execution |
| EXECUTION_TYPE | VARCHAR2(128) | | Type of the last execution. This information is optional for single-execution tasks. |
| EXECUTION_TYPE# | NUMBER | | Reserved for internal use |
| EXECUTION_DESCRIPTION | VARCHAR2(256) | | Optional description of the last execution |
| EXECUTION_START | DATE | | Execution start date and time of the task |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXECUTION_END | DATE | | Execution end date and time of the task |
| STATUS | VARCHAR2(11) | | Current operational status of the task:<br>• INITIAL - Initial state of the task; no recommendations are present<br>• EXECUTING - Task is currently running<br>• INTERRUPTED - Task analysis was interrupted by the user. Recommendation data, if present, can be viewed and reported at this time.<br>• COMPLETED - Task successfully completed the analysis operation. Recommendation data can be viewed and reported.<br>• ERROR - An error occurred during the analysis operation. Recommendations, if present, can be viewed and reported at this time. |
| STATUS_MESSAGE | VARCHAR2(4000) | | Informational message provided by the advisor, regarding the status |
| PCT_COMPLETION_TIME | NUMBER | | Percent completion, in terms of time, of the task when it is executing |
| PROGRESS_METRIC | NUMBER | | Metric that measures the progress of the task in terms of quality. Each advisor may have its own metric. |
| METRIC_UNITS | VARCHAR2(64) | | Unit of the metric used to measure progress |
| ACTIVITY_COUNTER | NUMBER | | Counter that is updated frequently by the advisor, denoting that useful work is being performed |
| RECOMMENDATION_COUNT | NUMBER | | Number of recommendations produced |
| ERROR_MESSAGE | VARCHAR2(4000) | | Informational message or an error message indicating the current operation or condition |
| SOURCE | VARCHAR2(128) | | Optional name that identifies the creator of the task |
| HOW_CREATED | VARCHAR2(30) | | Optional task or template on which the object was based |
| READ_ONLY | VARCHAR2(5) | | Indicates whether the task is read-only (TRUE) or not (FALSE) |
| SYSTEM_TASK | VARCHAR2(5) | | Indicates whether the task is a system task (TRUE) or not (FALSE). The automatic SQL tuning task, SYS_AUTO_SQL_TUNING_TASK, is one example of a system task. |
| ADVISOR_ID | NUMBER | NOT NULL | Unique identifier for the advisor |
| STATUS# | NUMBER | | Reserved for internal use |
| DBID | NUMBER | | The database ID that the task was using |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBNAME | VARCHAR2(9) | | The name of the database that the task was analyzing |
| DBVERSION | VARCHAR2(17) | | The version of the database that the task was analyzing |
| ANALYSIS_VERSION | VARCHAR2(17) | | The version of the database that executed the task |
| BEGIN_SNAP_ID | NUMBER | | The snapshot ID that starts the analysis period |
| BEGIN_TIME | TIMESTAMP(3) | | The SYSDATE at the time the BEGIN_SNAP_ID was taken |
| END_SNAP_ID | NUMBER | | The snapshot ID that ends the analysis period |
| END_TIME | TIMESTAMP(3) | | The SYSDATE at the time the END_SNAP_ID was taken |
| REQUESTED_ANALYSIS | VARCHAR2(8) | | The type of ADDM analysis that was requested before task execution, as follows:<br>DATABASE - global ADDM<br>INSTANCE - local ADDM<br>PARTIAL - continental ADDM |
| ACTUAL_ANALYSIS | VARCHAR2(8) | | The type of ADDM analysis that was actually performed when the task was executed (either DATABASE, INSTANCE, or PARTIAL) |
| DATABASE_TIME | NUMBER | | The total database time accumulated in the analysis period (and analyzed instances) in microseconds |
| ACTIVE_SESSIONS | NUMBER | | The average active sessions during the analysis period (and analyzed sessions) |
| METER_LEVEL | VARCHAR2(6) | | Reserved for future use |
| FDG_COUNT | NUMBER | | The number of findings for the ADDM task, which will appear in the list of findings in a default ADDM report |

**See Also:**

"USER_ADDM_TASKS"

# DBA_ADVISOR_ACTIONS

DBA_ADVISOR_ACTIONS displays information about the actions associated with all recommendations in the database. Each action is specified by the COMMAND and ATTR1 through ATTR6 columns. Each command defines how the attribute columns will be used.

### Related View

USER_ADVISOR_ACTIONS displays information about the actions associated with the recommendations owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |
| REC_ID | NUMBER | NOT NULL | Recommendation associated with the action |
| ACTION_ID | NUMBER | NOT NULL | Unique identifier for the action |
| OBJECT_ID | NUMBER | | Object associated with the action |
| COMMAND | VARCHAR2(64) | | Command to be executed<br>**See Also:** DBA_ADVISOR_COMMANDS for a list of commands |
| COMMAND_ID | NUMBER | NOT NULL | ID of the command to be executed<br>**See Also:** DBA_ADVISOR_COMMANDS for a list of commands |
| FLAGS | NUMBER | | Advisor-specific flags |
| ATTR1 | VARCHAR2(4000) | | Parameters defining the command |
| ATTR2 | VARCHAR2(4000) | | Parameters defining the command |
| ATTR3 | VARCHAR2(4000) | | Parameters defining the command |
| ATTR4 | VARCHAR2(4000) | | Parameters defining the command |
| ATTR5 | CLOB | | Parameters defining the command; to be used if the text is significantly large (for example, a SQL statement defining a materialized view) |
| ATTR6 | CLOB | | Parameters defining the command; to be used if the text is significantly large (for example, a SQL statement defining a materialized view) |
| NUM_ATTR1 | NUMBER | | General numeric attribute |
| NUM_ATTR2 | NUMBER | | General numeric attribute |
| NUM_ATTR3 | NUMBER | | General numeric attribute |
| NUM_ATTR4 | NUMBER | | General numeric attribute |
| NUM_ATTR5 | NUMBER | | General numeric attribute |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MESSAGE | VARCHAR2(4000) | | Message associated with the action |
| FILTERED | VARCHAR2(1) | | A value of Y means that the row in the view was filtered out by a directive (or a combination of directives). A value of N means that the row was not filtered. |

**See Also:**

"USER_ADVISOR_ACTIONS"

# DBA_ADVISOR_COMMANDS

DBA_ADVISOR_COMMANDS displays information about the commands used by all advisors in the database for specifying recommendation actions. In addition to the set of commands in the COMMAND column of V$SESSION, the following additional commands are defined:

- RUN ADVISOR
- CHECK EXECUTION PLAN
- ALTER PARAMETER
- ENABLE TRACE

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMMAND_ID | NUMBER | | Identifier of the command |
| COMMAND_NAME | VARCHAR2(64) | | Name of the command |

# DBA_ADVISOR_DEF_PARAMETERS

DBA_ADVISOR_DEF_PARAMETERS displays all default task parameters and their current values in the database. When a task or object is created, the parameters and their values are copied into the private parameter table.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADVISOR_NAME | VARCHAR2(128) | NOT NULL | Name of the advisor that supports the parameter |
| PARAMETER_NAME | VARCHAR2(128) | NOT NULL | Name of the parameter |
| PARAMETER_VALUE | VARCHAR2(4000) | NOT NULL | Value of the parameter. Numeric parameter values are converted to a string equivalent.<br>Possible keywords as values:<br>• ALL<br>• UNLIMITED<br>• UNUSED |
| PARAMETER_TYPE | VARCHAR2(10) | | Datatype of the parameter:<br>• NUMBER - Numeric value |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • STRING - String value. If the string contains special characters, then it will be enclosed in single quotes.<br>• STRINGLIST - Comma-separated list of string elements. If a string element contains a comma or other special characters, then the element will be enclosed in single quotes.<br>• TABLE - Single table reference. A reference will contain a schema name, followed by an optional table name.<br>If the table name is omitted or is the character %, then the table name is interpreted as a wildcard. SQL quoted identifiers are supported.<br>• TABLELIST - List of one or more comma-separated table references. A reference will contain schema name, followed by an optional table name.<br>If the table name is omitted or is the character %, then the table name is interpreted as a wildcard. SQL quoted identifiers are supported. |
| IS_DEFAULT | VARCHAR2(1) | | Indicates whether the parameter value is set to the advisor's default value (Y) or not (N) |
| IS_OUTPUT | VARCHAR2(1) | | Indicates whether the task execution process sets the parameter value (Y) or not (N) |
| IS_MODIFIABLE_ANYTIME | VARCHAR2(1) | | Indicates whether the parameter value can be modified when the task is not in its initial state (Y) or not (N) |
| IS_SYSTEM_TASK_ONLY | VARCHAR2(1) | | Indicates whether the task is a system task (Y) or not (N) |
| DESCRIPTION | VARCHAR2(4000) | | Optional description of the parameter |
| EXECUTION_TYPE | VARCHAR2(128) | | Type of the last execution. This information is optional for single-execution tasks. |

# DBA_ADVISOR_DEFINITIONS

DBA_ADVISOR_DEFINITIONS displays the properties of all advisors in the database. The view contains one row for each task, representing the current state of the task as well as execution-specific data such as progress monitoring and completion status.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADVISOR_ID | NUMBER | NOT NULL | Unique identifier for the advisor |
| ADVISOR_NAME | VARCHAR2(128) | NOT NULL | Name of the advisor |
| PROPERTY | NUMBER | NOT NULL | Properties:<br>• Bit 0: - Indicates whether the advisor runs in COMPREHENSIVE mode (1) or not (0) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • Bit 1: - Indicates whether the advisor runs in LIMITED mode (1) or not (0)<br>• Bit 2: - Indicates whether the advisor is resumable (1) or not (0)<br>• Bit 3: - Indicates whether the advisor accepts user directives (1) or not (0) |

# DBA_ADVISOR_DIR_DEFINITIONS

DBA_ADVISOR_DIR_DEFINITIONS provides a definition of the base directive.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ID | NUMBER | NOT NULL | Unique id for directive. The directive management engine automatically generates ID numbers. The identifier is unique among all directives regardless of the domain name and directive name. |
| ADVISOR_ID | NUMBER | NOT NULL | Identifier number of the owner advisor. |
| ADVISOR_NAME | VARCHAR2(128) | NOT NULL | The name of the advisor to which this directive belongs. |
| DIRECTIVE_NAME | VARCHAR2(128) | NOT NULL | Any value that further classifies this directive within a domain. The domain and the name form a unique key for the directive. |
| DOMAIN_NAME | VARCHAR2(128) | NOT NULL | Domain or namespace name. |
| DESCRIPTION | VARCHAR2(256) | NOT NULL | An optional description that documents the purpose of the directive. |
| TYPE | NUMBER | NOT NULL | Further describes the use of the directive. Possible values are:<br>1. **Filter** - An Xpath filter<br>2. **Single Value** - Evaluation returns a single string value<br>3. **Multiple Values** - Evaluation returns one to many string values<br>4. **Conditional** - Evaluation returns a single value based on an input key, similar to a CASE or SWITCH statement |
| TYPE_NAME | VARCHAR2(15) | | A decoded version of the TYPE column. |
| TASK_STATUS | VARCHAR2(9) | | The status of the directive instances when a task is not in its initial state. Possible values are:<br>• IMMUTABLE<br>• MUTABLE |
| INSTANCES | VARCHAR2(8) | | Indicates whether a directive will permit multiple instances. Possible values are:<br>• SINGLE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • MULTIPLE |
| METADATA | CLOB | NOT NULL | A DTD that is used to process the directive. |

# DBA_ADVISOR_DIR_INSTANCES

DBA_ADVISOR_DIR_INSTANCES provides information about all global instances for a directive.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DIRECTIVE_ID | NUMBER | NOT NULL | Unique id for directive. The directive management engine automatically generates ID numbers. |
| INSTANCE_ID | NUMBER | NOT NULL | Unique id for the directive instance. The directive management engine automatically generates ID numbers. |
| INSTANCE_NAME | VARCHAR2(128) | NOT NULL | A user-assigned name for the directive instance. |
| DATA | CLOB | NOT NULL | An XML document that gives meaningful default values for all parts of the directive. |

# DBA_ADVISOR_DIR_TASK_INST

DBA_ADVISOR_DIR_TASK_INST provides information about all task directive instances.

### Related View

USER_ADVISOR_DIR_TASK_INST provides information about all task directive instances owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DIRECTIVE_ID | NUMBER | NOT NULL | Unique id for directive. The directive management engine automatically generates ID numbers. |
| SEQ_ID | NUMBER | NOT NULL | Unique id for the directive instance. The directive management engine automatically generates ID numbers. |
| INSTANCE_NAME | VARCHAR2(128) | NOT NULL | A user-assigned name for the directive instance. |
| USERNAME | VARCHAR2(128) | NOT NULL | Database user who owns the task instance. |
| TASK_ID | NUMBER | | An advisor task identifier to which the directive instance is associated |
| TASK_NAME | VARCHAR2(128) | | An advisor task to which the directive instance is associated. |
| DATA | CLOB | NOT NULL | An XML document that gives meaningful default values for all parts of the directive. |

**See Also:**

"USER_ADVISOR_DIR_TASK_INST"

# DBA_ADVISOR_EXEC_PARAMETERS

DBA_ADVISOR_EXEC_PARAMETERS displays the parameter values used for past executions of tasks. It is more useful for advisors supporting multiple executions, such as SQL Performance Analyzer, where a parameter can have different values for different executions.

### Related View

USER_ADVISOR_EXEC_PARAMETERS displays the parameter values used for past executions of tasks owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | NOT NULL | Name of the task execution with which this entry (row) is associated |
| EXECUTION_TYPE | VARCHAR2(128) | | Type of the last execution. This information is optional for single-execution tasks. |
| PARAMETER_NAME | VARCHAR2(128) | NOT NULL | Name of the parameter |
| PARAMETER_VALUE | VARCHAR2(4000) | | Value of the parameter. Numeric parameter values are converted to a string equivalent. |
| PARAMETER_TYPE | VARCHAR2(10) | | Datatype of the parameter (see DBA_ADVISOR_PARAMETERS) |
| IS_DEFAULT | VARCHAR2(1) | | Indicates whether the parameter value is set to the advisor's default value (Y) or not (N) |
| IS_OUTPUT | VARCHAR2(1) | | Indicates whether the task execution process sets the parameter value (Y) or not (N) |
| IS_MODIFIABLE_ANYTIME | VARCHAR2(1) | | Indicates whether the parameter value can be modified when the task is not in its initial state (Y) or not (N) |
| DESCRIPTION | VARCHAR2(4000) | | Optional description of the parameter |
| PARAMETER_FLAGS | NUMBER | NOT NULL | Reserved for internal use |
| PARAMETER_TYPE# | NUMBER | NOT NULL | Reserved for internal use |

**See Also:**

"USER_ADVISOR_EXEC_PARAMETERS"

# DBA_ADVISOR_EXECUTION_TYPES

DBA_ADVISOR_EXECUTION_TYPES displays possible execution action for a given advisor. Only advisors that support multiple executions of their tasks have entries in this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADVISOR_NAME | VARCHAR2(128) | NOT NULL | Name of the advisor |
| EXECUTION_TYPE | VARCHAR2(128) | NOT NULL | Execution type supported by the advisor |
| EXECUTION_DESCRIPTION | VARCHAR2(4000) | | Optional description of the execution type |

# DBA_ADVISOR_EXECUTIONS

DBA_ADVISOR_EXECUTIONS displays metadata information for task executions. For example, the SQL Performance Analyzer creates a minimum of three executions to perform a change impact analysis on a SQL workload. The first one collects performance data for the version of the workload before the change, the second one collects data for the version of the workload after the change, and the third one performs impact analysis. All of these executions belong to the same task and are grouped into this view. Similarly, the automatic SQL tuning task, SYS_AUTO_SQL_TUNING_TASK, creates a new execution for each tuning run.

### Related View

USER_ADVISOR_EXECUTIONS displays metadata information for task executions owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | NOT NULL | Name of the task execution with which this entry (row) is associated |
| EXECUTION_ID | NUMBER | NOT NULL | Execution ID |
| DESCRIPTION | VARCHAR2(256) | | User-supplied description of the task |
| EXECUTION_TYPE | VARCHAR2(128) | | Type of the last execution (optional for single-execution tasks) |
| EXECUTION_TYPE# | NUMBER | | Reserved for internal use |
| EXECUTION_START | DATE | | Execution start date and time |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXECUTION_LAST_MODIFIED | DATE | NOT NULL | Last modified date and time for the execution |
| EXECUTION_END | DATE | | Execution end date and time |
| ADVISOR_NAME | VARCHAR2(128) | | Advisor associated with the task |
| ADVISOR_ID | NUMBER | NOT NULL | Unique identifier for the advisor |
| STATUS | VARCHAR2(11) | | Current operational status of the task:<br>• EXECUTING<br>• COMPLETED<br>• INTERRUPTED<br>• CANCELLED<br>• FATAL ERROR |
| STATUS# | NUMBER | NOT NULL | Reserved for internal use |
| STATUS_MESSAGE | VARCHAR2(4000) | | Informational message provided by the advisor regarding the status |
| ERROR_MESSAGE | VARCHAR2(4000) | | Informational message or an error message indicating the current operation or condition |

**See Also:**

"USER_ADVISOR_EXECUTIONS"

# DBA_ADVISOR_FDG_BREAKDOWN

DBA_ADVISOR_FDG_BREAKDOWN describes the contribution from the different instances to the findings for each ADDM task. This view is populated only with ADDM tasks that are analyzing multiple instances (that is, the ACTUAL_ANALYSIS column in the task's row in DBA_ADDM_TASKS is set to DATABASE or PARTIAL).

### Related View

USER_ADVISOR_FDG_BREAKDOWN describes the contribution from the different instances to the findings for each ADDM task owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER | NOT NULL | Unique identifier of the task (see DBA_ADVISOR_TASKS and DBA_ADDM_TASKS) |
| FINDING_ID | NUMBER | NOT NULL | Identifier of the finding to which this breakdown applies (see DBA_ADVISOR_FINDINGS and DBA_ADDM_FINDINGS) |
| INSTANCE_NUMBER | NUMBER | NOT NULL | The number of the instance for the breakdown |
| IMPACT | NUMBER | | The database time (in microseconds) of the finding in the instance |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PERC_IMPACT | NUMBER | | Percentage of the contribution of the instance to the overall finding's impact |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |

**See Also:**

"USER_ADVISOR_FDG_BREAKDOWN"

# DBA_ADVISOR_FINDING_NAMES

DBA_ADVISOR_FINDING_NAMES provides a list of all finding names registered with the Advisor Framework.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ID | NUMBER | | ID of the finding name |
| ADVISOR_NAME | VARCHAR2(128) | NOT NULL | Advisor name |
| FINDING_NAME | VARCHAR2(4000) | | Finding name |

# DBA_ADVISOR_FINDINGS

DBA_ADVISOR_FINDINGS displays the findings discovered by all advisors in the database.

### Related View

USER_ADVISOR_FINDINGS displays the findings discovered by the advisors owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |
| FINDING_ID | NUMBER | NOT NULL | Identifier of the finding |
| FINDING_NAME | VARCHAR2(4000) | | Name of the finding |
| TYPE | VARCHAR2(11) | | Type of the finding:<br>• PROBLEM<br>• SYMPTOM<br>• ERROR |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • INFORMATION |
| TYPE_ID | NUMBER | NOT NULL | Numeric ID for the value in column TYPE |
| PARENT | NUMBER | NOT NULL | Identifier of the parent finding |
| OBJECT_ID | NUMBER | | Identifier of the associated object, if any |
| IMPACT_TYPE | VARCHAR2(4000) | | Impact of the finding on the system |
| IMPACT | NUMBER | | Impact value |
| MESSAGE | VARCHAR2(4000) | | Message describing the finding |
| MORE_INFO | VARCHAR2(4000) | | Additional info associated with the finding |
| FILTERED | VARCHAR2(1) | | A value of Y means that the row in the view was filtered out by a directive (or a combination of directives). A value of N means that the row was not filtered. |
| FLAGS | NUMBER | | For internal use only by advisor framework clients |

**See Also:**

"USER_ADVISOR_FINDINGS"

# DBA_ADVISOR_JOURNAL

DBA_ADVISOR_JOURNAL displays the journal entries for all tasks in the database.

### Related View

USER_ADVISOR_JOURNAL displays the journal entries for the tasks owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Identifier of the task or workload object |
| TASK_NAME | VARCHAR2(128) | | Name of the task or workload object |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |
| JOURNAL_ENTRY_SEQ | NUMBER | NOT NULL | Sequence number of the journal entry (unique for each task). This sequence number is used to order the data. |
| JOURNAL_ENTRY_TYPE | VARCHAR2(12) | | Type of the task:<br>• FATAL |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • ERROR<br>• WARNING<br>• INFORMATION<br>• INFORMATION[2 \| 3 \| 4 \| 5 \| 6] |
| JOURNAL_ENTRY | VARCHAR2(4000) | | Entry in the journal |

**See Also:**

"USER_ADVISOR_JOURNAL"

# DBA_ADVISOR_LOG

DBA_ADVISOR_LOG displays information about the current state of all tasks in the database, as well as execution-specific data such as progress monitoring and completion status. The view contains one row for each task.

### Related View

USER_ADVISOR_LOG displays information about the current state of the tasks owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_START | DATE | | Execution start date and time of the task |
| EXECUTION_END | DATE | | Execution end date and time of the task |
| STATUS | VARCHAR2(11) | | Current operational status of the task:<br>• INITIAL - Initial state of the task; no recommendations are present<br>• EXECUTING - Task is currently running<br>• COMPLETED - Task successfully completed the analysis operation. Recommendation data can be viewed and reported.<br>• INTERRUPTED - Task analysis was interrupted by the user. Recommendation data, if present, can be viewed and reported at this time.<br>• CANCELLED<br>• FATAL ERROR - A fatal error occurred during the analysis operation. All recommendation data is unusable. |
| STATUS_MESSAGE | VARCHAR2(4000) | | Informational message provided by the advisor regarding the status |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PCT_COMPLETION_TIME | NUMBER | | Percent completion, in terms of time, of the task when it is executing |
| PROGRESS_METRIC | NUMBER | | Metric that measures the progress of the task in terms of quality. Each advisor could have its own metric. |
| METRIC_UNITS | VARCHAR2(64) | | Unit of the metric used to measure progress |
| ACTIVITY_COUNTER | NUMBER | | Counter that is updated frequently by the advisor, denoting that useful work is being performed |
| RECOMMENDATION_COUNT | NUMBER | | Number of recommendations produced |
| ERROR_MESSAGE | VARCHAR2(4000) | | Informational message or an error message indicating the current operation or condition |

**See Also:**

"USER_ADVISOR_LOG"

# DBA_ADVISOR_OBJECT_TYPES

DBA_ADVISOR_OBJECT_TYPES displays information about the object types used by all advisors in the database. In addition to the regular database object types (such as TABLE and INDEX), the following types are defined:

- SYSTEM
- I/O
- SGA
- PGA
- SHARED POOL
- BUFFER CACHE
- LIBRARY CACHE
- PROCESS
- SESSION
- ENQUEUE
- LATCH
- ROLLBACK SEGMENT
- FILE
- PARAMETER

- CURSOR
- SQL
- SQL WORKLOAD

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_TYPE_ID | NUMBER | | Type identifier |
| OBJECT_TYPE | VARCHAR2(64) | | Type name |

# DBA_ADVISOR_OBJECTS

DBA_ADVISOR_OBJECTS displays information about the objects currently referenced by all advisors in the database. Each row in the view pertains to an object instantiation.

### Related View

USER_ADVISOR_OBJECTS displays information about the objects currently referenced by the advisors owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the object |
| OBJECT_ID | NUMBER | NOT NULL | Identifier of the object |
| TYPE | VARCHAR2(64) | | Name of the type |
| TYPE_ID | NUMBER | NOT NULL | Type identifier number |
| TASK_ID | NUMBER | NOT NULL | Task referencing the object |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |
| ATTR1 | VARCHAR2(4000) | | Attributes and identifier of the object |
| ATTR2 | VARCHAR2(4000) | | Attributes and identifier of the object |
| ATTR3 | VARCHAR2(4000) | | Attributes and identifier of the object |
| ATTR4 | CLOB | | Attributes and identifiers that cannot be expressed in the ATTR1, ATTR2, and ATTR3 columns |
| ATTR5 | VARCHAR2(4000) | | Attributes and identifier of the object |
| ATTR6 | RAW(2000) | | Attributes and identifier of the object |
| ATTR7 | NUMBER | | Attributes and identifier of the object |
| ATTR8 | NUMBER | | Attributes and identifier of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ATTR9 | NUMBER | | Attributes and identifier of the object |
| ATTR10 | NUMBER | | Attributes and identifier of the object |
| ATTR11 | NUMBER | | Attributes and identifier of the object |
| ATTR16 | VARCHAR2(4000) | | Attributes and identifier of the object |
| ATTR17 | VARCHAR2(4000) | | Attributes and identifier of the object |
| ATTR18 | VARCHAR2(4000) | | Attributes and identifier of the object |
| OTHER | CLOB | | Other attributes and identifiers of the object |

**Note:**

The definition of the ATTRn columns depends on the advisors that are using the object. For example, the SQL object type defines the attribute columns as follows:

- ATTR1 contains the SQL ID
- ATTR2 contains the SQL address (in the cursor cache)
- ATTR4 contains the SQL text

**See Also:**

"USER_ADVISOR_OBJECTS"

# DBA_ADVISOR_PARAMETERS

DBA_ADVISOR_PARAMETERS displays all task parameters and their current values in the database. This data is accessible by all tasks.

### Related View

USER_ADVISOR_PARAMETERS displays the task parameters and their current values for the tasks owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task or workload object |
| TASK_ID | NUMBER | NOT NULL | Unique identifier number of the task or workload object |
| TASK_NAME | VARCHAR2(128) | | Name of the task or workload object |
| PARAMETER_NAME | VARCHAR2(128) | NOT NULL | Name of the parameter |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETER_VALUE | VARCHAR2(4000) | NOT NULL | Value of the parameter. Numeric parameter values are converted to a string equivalent.<br>Possible keywords as values:<br>• ALL<br>• UNLIMITED<br>• UNUSED |
| PARAMETER_TYPE | VARCHAR2(10) | | Datatype of the parameter:<br>• NUMBER - Numeric value<br>• STRING - String value. If the string contains special characters, then it will be enclosed in single quotes.<br>• STRINGLIST - Comma-separated list of string elements. If a string element contains a comma or other special characters, then the element will be enclosed in single quotes.<br>• TABLE - Single table reference. A reference will contain a schema name, followed by an optional table name.<br>If the table name is omitted or is the character %, then the table name is interpreted as a wildcard. SQL quoted identifiers are supported.<br>• TABLELIST - List of one or more comma-separated table references. A reference will contain schema name, followed by an optional table name.<br>If the table name is omitted or is the character %, then the table name is interpreted as a wildcard. SQL quoted identifiers are supported. |
| IS_DEFAULT | VARCHAR2(1) | | Indicates whether the parameter value is set to the advisor's default value (Y) or not (N) |
| IS_OUTPUT | VARCHAR2(1) | | Indicates whether the task execution process sets the parameter value (Y) or not (N) |
| IS_MODIFIABLE_ANYTIME | VARCHAR2(1) | | Indicates whether the parameter value can be modified when the task is not in its initial state (Y) or not (N) |
| DESCRIPTION | VARCHAR2(4000) | | Optional description of the parameter |
| EXECUTION_TYPE | VARCHAR2(128) | | For advisors supporting multiple executions, the type of execution this parameter pertains to |

**See Also:**

"USER_ADVISOR_PARAMETERS"

# DBA_ADVISOR_RATIONALE

DBA_ADVISOR_RATIONALE displays information about the rationales for all recommendations in the database.

### Related View

USER_ADVISOR_RATIONALE displays information about the rationales for the recommendations owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |
| REC_ID | NUMBER | | Recommendation associated with the rationale |
| RATIONALE_ID | NUMBER | NOT NULL | Unique identifier for the rationale |
| IMPACT_TYPE | VARCHAR2(4000) | | Impact on the system due to the problem described in the rationale. The impact can be described in terms of time, cost, or % degradation. |
| IMPACT | NUMBER | | Calculated impact value |
| MESSAGE | VARCHAR2(4000) | | Message containing an overview of the rationale |
| OBJECT_ID | NUMBER | | Identifier of an object specified in the DBA_ADVISOR_OBJECTS view |
| TYPE | VARCHAR2(30) | | Type of the rationale; defines what data exists in the attribute columns and how to interpret it:<br>• TEXT - Text sentence for descriptive messages. The ATTR1 column contains the text.<br>• CHART - Chart containing data to be displayed. The ATTR1 column contains the data. |
| ATTR1 | VARCHAR2(4000) | | Parameters defining the rationale |
| ATTR2 | VARCHAR2(4000) | | Parameters defining the rationale |
| ATTR3 | VARCHAR2(4000) | | Parameters defining the rationale |
| ATTR4 | VARCHAR2(4000) | | Parameters defining the rationale |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ATTR5 | CLOB | | Parameters defining the rationale |

**See Also:**

"USER_ADVISOR_RATIONALE"

# DBA_ADVISOR_RECOMMENDATIONS

DBA_ADVISOR_RECOMMENDATIONS displays the results of an analysis of all recommendations in the database. A recommendation can have multiple actions associated with it. Actions are described in the DBA_ADVISOR_ACTIONS view. A recommendation also points to a set of rationales that present a justification/reasoning for that recommendation. These rationales are in the DBA_ADVISOR_RATIONALE view.

### Related View

USER_ADVISOR_RECOMMENDATIONS displays the results of an analysis of the recommendations owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| REC_ID | NUMBER | NOT NULL | Unique identifier of the recommendation |
| TASK_ID | NUMBER | NOT NULL | Task that owns the recommendation |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| EXECUTION_NAME | VARCHAR2(128) | | The name of the task execution with which this entry (row) is associated |
| FINDING_ID | NUMBER | | Unique identifier of the finding |
| TYPE | VARCHAR2(30) | | Type of the recommendation |
| RANK | NUMBER | | Ranking, in terms of importance, within the set of recommendations generated for the task |
| PARENT_REC_IDS | VARCHAR2(4000) | | Comma-separated list of the recommendation IDs of the parent recommendations. If this column is nonzero, then the recommendation depends on the parents, and cannot be accepted if the parents are not accepted. |
| BENEFIT_TYPE | VARCHAR2(4000) | | Describes the benefit obtained by carrying out the recommendation<br>If there is a set of parents for the recommendation, then the benefit is the cumulative benefit (the improvement in system performance when this and all prior parent recommendations are accepted).<br>If there are no parents, then this is the improvement when the recommendation is |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | accepted, independent of other recommendations. |
| BENEFIT | NUMBER | | Calculated benefit value |
| ANNOTATION_STATUS | VARCHAR2(11) | | When a task is complete, the recommendations are marked ACCEPT. The status can be changed later using the MARK_RECOMMENDATION procedure:<br>• ACCEPT - Current recommendation is ready to implement. This recommendation can also be used as advice for future analysis operations.<br>• REJECT - Current recommendation is not acceptable to the user, and therefore will be excluded from any implementation scripts. This recommendation can also be used as advice for future analysis operations.<br>• IGNORE - Though not rejected, the current recommendation will be ignored when generating scripts and will never be used as advice to future analysis operations. |
| FLAGS | NUMBER | | Advisor-specific flags |
| FILTERED | VARCHAR2(1) | | A value of Y means that the row in the view was filtered out by a directive (or a combination of directives). A value of N means that the row was not filtered. |

**See Also:**

"USER_ADVISOR_RECOMMENDATIONS"

# DBA_ADVISOR_SQLA_REC_SUM

DBA_ADVISOR_SQLA_REC_SUM displays recommendation rollup information for all workload objects in the database after an Access Advisor analysis operation.

### Related View

USER_ADVISOR_SQLA_REC_SUM displays recommendation rollup information for the workload objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| REC_ID | NUMBER | | Identifier of the recommendation |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TOTAL_STMTS | NUMBER | | Total number of statements processed during analysis |
| TOTAL_PRECOST | NUMBER | | Total cost of executing the statements in which the recommended object will be used, before the recommendations |
| TOTAL_POSTCOST | NUMBER | | Total cost of executing the statements in which the recommended object will be utilized, after the recommendations have been implemented |

**See Also:**

"USER_ADVISOR_SQLA_REC_SUM"

# DBA_ADVISOR_SQLA_TABLES

DBA_ADVISOR_SQLA_TABLES displays cross references between the workload statements and the tables referenced in the statement.

### Related View

USER_ADVISOR_SQLA_TABLES displays cross references between the workload statements and the tables referenced in the statement for the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the workload object |
| TASK_ID | NUMBER | | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| SQL_ID | VARCHAR2(13) | | SQL identifier of the parent cursor in the library cache |
| STMT_ID | NUMBER | | Statement ID |
| TABLE_OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Table name |

**See Also:**

"USER_ADVISOR_SQLA_TABLES"

# DBA_ADVISOR_SQLA_WK_MAP

DBA_ADVISOR_SQLA_WK_MAP displays the workload references for all tasks in the database. Workload references are necessary to allow the SQL Access Advisor to find required workload data.

### Related View

USER_ADVISOR_SQLA_WK_MAP displays the workload references for the tasks owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| WORKLOAD_ID | NUMBER | | Unique identifier of the workload object |
| WORKLOAD_NAME | VARCHAR2(128) | | Name of the workload |
| IS_STS | NUMBER | | Type of workload source:<br>• 0 - SQL workload object<br>• 1 - SQL Tuning Set |

**See Also:**

"USER_ADVISOR_SQLA_WK_MAP"

## DBA_ADVISOR_SQLA_WK_STMTS

DBA_ADVISOR_SQLA_WK_STMTS displays information about all workload objects in the database after an Access Advisor analysis operation.

### Related View

USER_ADVISOR_SQLA_WK_STMTS displays information about the workload objects owned by the current user after an Access Advisor analysis operation. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| TASK_ID | NUMBER | NOT NULL | Unique identifier of the task |
| SQLSET_ID | NUMBER | | ID of the SQL tuning set for the statement |
| SQLSET_NAME | VARCHAR2(128) | | Name of the SQL tuning set for the statement |
| WORKLOAD_NAME | VARCHAR2(128) | | Name of the workload |
| SQL_ID | VARCHAR(13) | NOT NULL | Generated identifier of the statement |
| SQL_SEQ | NUMBER | NOT NULL | SQL sequence |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Numerical representation of the SQL plan for the cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | are the same (rather than comparing the two plans line-by-line). |
| PARSING_SCHEMA_NAME | VARCHAR2(128) | | Schema name that was used to originally build this child cursor |
| USERNAME | VARCHAR2(128) | | Name of the user executing the statement |
| MODULE | VARCHAR2(64) | | Name of the module issuing the statement |
| ACTION | VARCHAR2(64) | | Module action for the statement |
| CPU_TIME | NUMBER | | Total CPU count (in seconds) of the executing statement |
| BUFFER_GETS | NUMBER | | Total number of buffer gets for the statement |
| DISK_READS | NUMBER | | Total disk-read I/O count for the statement |
| ELAPSED_TIME | NUMBER | | Total elapsed time (in seconds) of the executing statement |
| ROWS_PROCESSED | NUMBER | | Total number of rows processed by the statement |
| EXECUTIONS | NUMBER | | Total number of times the statement was executed |
| FIRST_LOAD_TIME | DATE | | Load time of parent cursor |
| LAST_EXECUTION_DATE | DATE | | Date on which the statement was last executed |
| PRIORITY | NUMBER | | Business importance of the statement:<br>• 1 - High<br>• 2 - Medium<br>• 3 - Low |
| COMMAND_TYPE | NUMBER | | Type of the command |
| STAT_PERIOD | NUMBER | | Unused |
| ACTIVE_STAT_PERIOD | NUMBER | | Effective period of time (in seconds) during which the SQL statement was active |
| SQL_TEXT | CLOB | | Text of the SQL statement |
| PRECOST | NUMBER | | Cost of executing the statement in the workload prior to the recommendations |
| POSTCOST | NUMBER | | Cost of executing the statement in the workload after the recommendations |
| IMPORTANCE | NUMBER | | Advisor-calculated importance value |
| REC_ID | NUMBER | | Associated recommendation identifier |
| VALIDATED | NUMBER | | Indicates whether the statement is valid for analysis:<br>• 0 - Statement will not be analyzed by the EXECUTE_TASK procedure. Typically, the |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | statement references one or more tables that do not have valid statistics. To correct this problem, ensure that the tables have valid statistics and execute the RESET_SQLWKLD procedure on the current workload.<br>• 1- Statement is eligible for analysis by the EXECUTE_TASK procedure |

**See Also:**

"USER_ADVISOR_SQLA_WK_STMTS"

# DBA_ADVISOR_SQLPLANS

DBA_ADVISOR_SQLPLANS displays the different SQL execution plans generated as part of an advisor analysis.

### Related View

USER_ADVISOR_SQLPLANS displays the different SQL execution plans owned by the current user generated as part of an advisor analysis.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_NAME | VARCHAR2(128) | | Advisor task name in which the SQL plan was generated (see DBA_ADVISOR_TASKS) |
| TASK_ID | NUMBER(38) | NOT NULL | Advisor task ID in which the SQL plan was generated (see DBA_ADVISOR_TASKS) |
| EXECUTION_NAME | VARCHAR2(128) | NOT NULL | Advisor task execution in which the SQL plan was generated (see DBA_ADVISOR_EXECUTIONS) |
| SQL_ID | VARCHAR2(13) | NOT NULL | Identifier for the relevant SQL statement |
| OBJECT_ID | NUMBER(38) | NOT NULL | Advisor object ID identifying the relevant SQL statement (see DBA_ADVISOR_OBJECTS) |
| ATTRIBUTE | VARCHAR2(34) | | Text string identifying the type of the execution plan. The following values are used by the SQL Tuning Advisor:<br>• Original - Original plan of the query<br>• Original with adjusted cost - Same as Original but with adjusted cost<br>• Using SQL profile - Plan with SQL profile applied<br>• Using new indices - Plan with indexes applied |
| STATEMENT_ID | VARCHAR2(30) | | Optional statement identifier specified in the EXPLAIN PLAN statement |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Numerical representation of the execution plan |
| PLAN_ID | NUMBER | NOT NULL | Plan identifier |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TIMESTAMP | DATE | | Date and time when the EXPLAIN PLAN statement was issued |
| REMARKS | VARCHAR2(4000) | | Place for comments that can be added to the steps of the execution plan |
| OPERATION | VARCHAR2(30) | | Name of the operation performed at this step |
| OPTIONS | VARCHAR2(255) | | Options used for the operation performed at this step |
| OBJECT_NODE | VARCHAR2(128) | | Name of the database link used to reference the object |
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object |
| OBJECT_ALIAS | VARCHAR2(261) | | Object alias |
| OBJECT_INSTANCE | NUMBER(38) | | Numbered position of the object name in the original SQL statement |
| OBJECT_TYPE | VARCHAR2(30) | | Descriptive modifier that further describes the type of object |
| OPTIMIZER | VARCHAR2(255) | | Current mode of the optimizer |
| SEARCH_COLUMNS | NUMBER | | Number of index columns with start and stop keys (that is, the number of columns with matching predicates) |
| ID | NUMBER(38) | NOT NULL | Identification number for this step in the execution plan |
| PARENT_ID | NUMBER(38) | | ID of the next step that operates on the results of this step |
| DEPTH | NUMBER(38) | | Depth |
| POSITION | NUMBER(38) | | Order of processing for steps with the same parent ID |
| COST | NUMBER(38) | | Cost of the current operation estimated by the cost-based optimizer (CBO) |
| CARDINALITY | NUMBER(38) | | Number of rows returned by the current operation (estimated by the CBO) |
| BYTES | NUMBER(38) | | Number of bytes returned by the current operation |
| OTHER_TAG | VARCHAR2(255) | | Describes the function of the SQL text in the OTHER column. Values for OTHER_TAG are:<br>• SERIAL - SQL is the text of a locally-executed, serial query plan. Currently, SQL is not loaded in OTHER for this case.<br>• SERIAL_FROM_REMOTE - SQL text shown in the OTHER column will be executed at a remote site |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • PARALLEL_COMBINED_WITH_PARENT - Parent of this operation is a DFO that performs both operations in the parallel execution plan<br>• PARALLEL_COMBINED_WITH_CHILD - Child of this operation is a DFO that performs both operations in the parallel execution plan.<br>• PARALLEL_TO_SERIAL - SQL text shown in the OTHER column is the top-level of the parallel plan.<br>• PARALLEL_TO_PARALLEL - SQL text shown in the OTHER column is executed and output in parallel<br>• PARALLEL_FROM_SERIAL - Operation consumes data from a serial operation and outputs it in parallel |
| PARTITION_START | VARCHAR2(255) | | Start partition of a range of accessed partitions |
| PARTITION_STOP | VARCHAR2(255) | | Stop partition of a range of accessed partitions |
| PARTITION_ID | NUMBER(38) | | Step that has computed the pair of values of the PARTITION_START and PARTITION_STOP columns |
| OTHER | LONG | | Information about parallel execution servers and parallel queries |
| DISTRIBUTION | VARCHAR2(30) | | Distribution method |
| CPU_COST | NUMBER(38) | | User-defined CPU cost |
| IO_COST | NUMBER(38) | | User-defined I/O cost |
| TEMP_SPACE | NUMBER(38) | | Temporary space usage of the operation (sort or hash-join) as estimated by the CBO |
| ACCESS_PREDICATES | VARCHAR2(4000) | | Predicates used to locate rows in an access structure. For example, start or stop predicates for an index range scan. |
| FILTER_PREDICATES | VARCHAR2(4000) | | Predicates used to filter rows before producing them |
| PROJECTION | VARCHAR2(4000) | | Expressions produced by the operation |
| TIME | NUMBER(38) | | Elapsed time (in seconds) of the operation as estimated by the CBO |
| QBLOCK_NAME | VARCHAR2(128) | | Name of the query block |
| OTHER_XML | CLOB | | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML because it allows multiple pieces of information to be stored, including the following:<br>• Name of the schema against which the query was parsed |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan<br>• Indication of whether or not dynamic statistics were used to produce the plan<br>• The outline data, a set of optimizer hints that can be used to regenerate the same plan |

**See Also:**

"USER_ADVISOR_SQLPLANS"

# DBA_ADVISOR_SQLSTATS

DBA_ADVISOR_SQLSTATS displays execution statistics for the test-execution of different SQL plans during the advisor analysis.

### Related View

USER_ADVISOR_SQLSTATS displays execution statistics owned by the current user for the test-execution of different SQL plans during the advisor analysis.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_NAME | VARCHAR2(128) | | Advisor task name in which the SQL statement was executed (see DBA_ADVISOR_TASKS) |
| TASK_ID | NUMBER(38) | NOT NULL | Advisor task ID in which the SQL statement was executed (see DBA_ADVISOR_TASKS) |
| EXECUTION_NAME | VARCHAR2(128) | NOT NULL | Advisor task execution in which the SQL statement was executed (see DBA_ADVISOR_EXECUTIONS) |
| EXECUTION_TYPE | VARCHAR2(128) | | Type of the advisor task execution in which the SQL statement was executed (see DBA_ADVISOR_EXECUTIONS) |
| OBJECT_ID | NUMBER(38) | NOT NULL | Advisor object ID identifying the relevant SQL statement (see DBA_ADVISOR_OBJECTS) |
| PLAN_ID | NUMBER | NOT NULL | Plan ID number generated to uniquely identify a plan for a particular SQL statement (foreign key to DBA_ADVISOR_SQLPLANS) |
| SQL_ID | VARCHAR2(13) | NOT NULL | Identifier for the SQL statement executed |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Hash value of the SQL execution plan |
| ATTR1 | NUMBER | | Reserved for internal use |
| CON_DBID | NUMBER | | The database ID of the pluggable database (PDB) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARSE_TIME | NUMBER | | Parse time (in microseconds) measured for the SQL |
| ELAPSED_TIME | NUMBER | | Elapsed time (in microseconds) to execute the SQL and fetch all of its rows, after parsing |
| CPU_TIME | NUMBER | | CPU time (in microseconds) to execute the SQL and fetch all of its rows, after parsing |
| USER_IO_TIME | NUMBER | | I/O time (in microseconds) to execute the SQL and fetch all of its rows, after parsing |
| BUFFER_GETS | NUMBER | | Number of buffer gets measured for executing the SQL and fetching all of its rows |
| DISK_READS | NUMBER | | Number of disk reads measured for executing the SQL and fetching all of its rows |
| DIRECT_WRITES | NUMBER | | Number of direct writes measured for executing the SQL and fetching all of its rows |
| PHYSICAL_READ_REQUESTS | NUMBER | | Number of physical read I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS | NUMBER | | Number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_READ_BYTES | NUMBER | | Number of bytes read from disks by the monitored SQL |
| PHYSICAL_WRITE_BYTES | NUMBER | | Number of bytes written to disks by the monitored SQL |
| ROWS_PROCESSED | NUMBER | | Number of rows returned by the SQL execution |
| FETCHES | NUMBER | | Number of fetches for the SQL execution |
| EXECUTIONS | NUMBER | | Execution count for the SQL. This column will always have a value of `1` or `0`. |
| END_OF_FETCH_COUNT | NUMBER | | Indicates whether the SQL was executed to end-of-fetch (`1`) or not (`0`) |
| OPTIMIZER_COST | NUMBER | | Optimizer cost for the execution plan |
| OTHER | CLOB | | Reserved for future use |
| TESTEXEC_TOTAL_EXECS | NUMBER | | Total number of executions during test execute |
| IO_INTERCONNECT_BYTES | NUMBER | | Number of I/O bytes exchanged between Oracle Database and the storage system |
| TESTEXEC_FIRST_EXEC_IGNORED | VARCHAR2(1) | | Indicates whether the first execution in test execute is ignored (`Y`) or not (`N`) |
| CON_DBID | NUMBER | | The database ID of the PDB |

**See Also:**

"USER_ADVISOR_SQLSTATS"

# DBA_ADVISOR_SQLW_JOURNAL

DBA_ADVISOR_SQLW_JOURNAL displays the journal entries for all workload objects in the database.

### Related View

USER_ADVISOR_SQLW_JOURNAL displays the journal entries for the workload objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the workload |
| WORKLOAD_ID | NUMBER | NOT NULL | Identifier number of the workload object |
| WORKLOAD_NAME | VARCHAR2(128) | | Name of the workload object |
| JOURNAL_ENTRY_SEQ | NUMBER | NOT NULL | Sequence number of the journal entry (unique for each workload). The sequence number is used to order the data. |
| JOURNAL_ENTRY_TYPE | VARCHAR2(12) | | Type of the task:<br>• FATAL<br>• ERROR<br>• WARNING<br>• INFORMATION<br>• INFORMATION[2 \| 3 \| 4 \| 5 \| 6] |
| JOURNAL_ENTRY | VARCHAR2(4000) | | Entry in the journal |

**See Also:**

"USER_ADVISOR_SQLW_JOURNAL"

# DBA_ADVISOR_SQLW_PARAMETERS

DBA_ADVISOR_SQLW_PARAMETERS displays all workload parameters and their current values in the database.

### Related View

USER_ADVISOR_SQLW_PARAMETERS displays the workload parameters and their current values owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task or workload object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WORKLOAD_ID | NUMBER | NOT NULL | Unique identifier number of the workload object |
| WORKLOAD_NAME | VARCHAR2(128) | | Name of the workload object |
| PARAMETER_NAME | VARCHAR2(128) | NOT NULL | Name of the parameter |
| PARAMETER_VALUE | VARCHAR2(4000) | NOT NULL | Value of the parameter. Numeric parameter values are converted to a string equivalent.<br>Possible keywords as values:<br>• ALL<br>• UNLIMITED<br>• UNUSED |
| PARAMETER_TYPE | VARCHAR2(10) | | Datatype of the parameter:<br>• NUMBER - Numeric value<br>• STRING - String value. If the string contains special characters, then it will be enclosed in single quotes.<br>• STRINGLIST - Comma-separated list of string elements. If a string element contains a comma or other special characters, then the element will be enclosed in single quotes.<br>• TABLE - Single table reference. A reference contains a schema name, followed by an optional table name.<br>If the table name is omitted or is the character %, then the table name is interpreted as a wildcard. SQL quoted identifiers are supported.<br>• TABLELIST - List of one or more comma-separated table references. A reference contains a schema name, followed by an optional table name.<br>If the table name is omitted or is the character %, then the table name is interpreted as a wildcard. SQL quoted identifiers are supported. |
| DESCRIPTION | VARCHAR2(4000) | | Parameter description |

**See Also:**

"USER_ADVISOR_SQLW_PARAMETERS"

# DBA_ADVISOR_SQLW_STMTS

DBA_ADVISOR_SQLW_STMTS displays rows that correspond to all statements in the workload. All columns are guaranteed to be non-null.

### Related View

USER_ADVISOR_SQLW_STMTS displays rows that correspond to the statements in the workload owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the workload object |
| WORKLOAD_ID | NUMBER | NOT NULL | Unique identifier number of the workload object |
| WORKLOAD_NAME | VARCHAR2(128) | | Name of the workload |
| SQL_ID | NUMBER | NOT NULL | Generated identifier of the statement |
| HASH_VALUE | NUMBER | | Hash value for the parent statement in the cache |
| USERNAME | VARCHAR2(128) | | Name of the user executing the statement |
| MODULE | VARCHAR2(64) | | Name of the module issuing the statement |
| ACTION | VARCHAR2(64) | | Module action for the statement |
| CPU_TIME | NUMBER | | Total CPU count (in seconds) of the executing statement |
| BUFFER_GETS | NUMBER | | Total number of buffer gets for the statement |
| DISK_READS | NUMBER | | Total disk-read I/O count for the statement |
| ELAPSED_TIME | NUMBER | | Total elapsed time (in seconds) of the executing statement |
| ROWS_PROCESSED | NUMBER | | Total number of rows processed by the statement |
| EXECUTIONS | NUMBER | | Total number of times the statement was executed |
| OPTIMIZER_COST | NUMBER | | Cost of executing the statement in the workload prior to the recommendations |
| LAST_EXECUTION_DATE | DATE | | Date on which the statement was last executed |
| PRIORITY | NUMBER | | Priority of the statement:<br>• 1 - High<br>• 2 - Medium<br>• 3 - Low |
| COMMAND_TYPE | NUMBER | | Type of the command |
| STAT_PERIOD | NUMBER | | Unused |
| SQL_TEXT | CLOB | | Text of the SQL statement |
| VALID | NUMBER | | Indicates whether the statement is valid for analysis:<br>• 0 - Statement will not be analyzed by the EXECUTE_TASK procedure. Typically, the statement references one or more tables that do not have valid statistics. To correct this problem, ensure that the tables have valid statistics and execute the RESET_SQLWKLD procedure on the current workload.<br>• 1 - Statement is eligible for analysis by the EXECUTE_TASK procedure. |

**See Also:**

"USER_ADVISOR_SQLW_STMTS"

# DBA_ADVISOR_SQLW_SUM

DBA_ADVISOR_SQLW_SUM displays an aggregated picture of all SQLWkld workload objects in the database.

### Related View

USER_ADVISOR_SQLW_SUM displays an aggregated picture of the SQLWkld workload objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the workload object |
| WORKLOAD_ID | NUMBER | NOT NULL | Unique identifier number of the workload object |
| WORKLOAD_NAME | VARCHAR2(128) | | Unique name of the workload |
| DESCRIPTION | VARCHAR2(256) | | User-specified description of the workload |
| CREATE_DATE | DATE | NOT NULL | Date on which the workload object was created |
| MODIFY_DATE | DATE | NOT NULL | Date of last update to the current workload |
| NUM_SELECT_STMT | NUMBER | | Number of SELECT statements in the workload |
| NUM_UPDATE_STMT | NUMBER | | Number of UPDATE statements in the workload |
| NUM_DELETE_STMT | NUMBER | | Number of DELETE statements in the workload |
| NUM_INSERT_STMT | NUMBER | | Number of INSERT statements in the workload |
| NUM_MERGE_STMT | NUMBER | | Number of MERGE statements in the workload |
| SOURCE | VARCHAR2(128) | | Optional name that identifies the creator of the object |
| HOW_CREATED | VARCHAR2(30) | | Optional object or template on which the object was based |
| DATA_SOURCE | VARCHAR2(2000) | | Workload data source |
| READ_ONLY | VARCHAR2(5) | | Indicates whether or not the workload can be modified or deleted (TRUE) or not (FALSE) |

**See Also:**

"USER_ADVISOR_SQLW_SUM"

# DBA_ADVISOR_SQLW_TABLES

DBA_ADVISOR_SQLW_TABLES displays cross references between the workload statements and the tables referenced in the statement.

### Related View

USER_ADVISOR_SQLW_TABLES displays cross references between the workload statements and the tables referenced in the statement. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the workload object |
| WORKLOAD_ID | NUMBER | | Unique identifier number of the workload object |
| WORKLOAD_NAME | VARCHAR2(128) | | Name of the workload |
| SQL_ID | NUMBER | | Identifier of the statement |
| TABLE_OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |

**See Also:**

"USER_ADVISOR_SQLW_TABLES"

# DBA_ADVISOR_SQLW_TEMPLATES

DBA_ADVISOR_SQLW_TEMPLATES displays an aggregated picture of all SQLWkld template objects in the database.

### Related View

USER_ADVISOR_SQLW_TEMPLATES displays an aggregated picture of the SQLWkld template objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the workload object |
| WORKLOAD_ID | NUMBER | NOT NULL | Unique identifier number of the workload object |
| WORKLOAD_NAME | VARCHAR2(128) | | Unique name of the workload |
| DESCRIPTION | VARCHAR2(256) | | User-specified description of the workload |
| CREATE_DATE | DATE | NOT NULL | Date on which the workload object was created |
| MODIFY_DATE | DATE | NOT NULL | Date of last update to the current workload |
| SOURCE | VARCHAR2(128) | | Optional object or template on which the object was based |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| READ_ONLY | VARCHAR2(5) | | Indicates whether the workload template can be modified or deleted (TRUE) or not (FALSE) |

**See Also:**

"USER_ADVISOR_SQLW_TEMPLATES"

# DBA_ADVISOR_TASKS

DBA_ADVISOR_TASKS displays information about all tasks in the database. The view contains one row for each task. Each task has a name that is unique to the owner. Task names are just informational and no uniqueness is enforced within any other namespace.

### Related View

USER_ADVISOR_TASKS displays information about the tasks owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| DESCRIPTION | VARCHAR2(256) | | User-supplied description of the task |
| ADVISOR_NAME | VARCHAR2(128) | | Advisor associated with the task |
| CREATED | DATE | NOT NULL | Creation date of the task |
| LAST_MODIFIED | DATE | NOT NULL | Date on which the task was last modified |
| PARENT_TASK_ID | NUMBER | | Identifier of the parent task (if the task was created because of the recommendation of another task) |
| PARENT_RXEC_ID | NUMBER | | Identifier of the recommendation within the parent task that resulted in the creation of the task |
| LAST_EXECUTION | VARCHAR2(128) | | Name of the current or last task execution |
| EXECUTION_TYPE | VARCHAR2(128) | | Type of the last execution. This information is optional for single-execution tasks. |
| EXECUTION_TYPE# | NUMBER | | Reserved for internal use |
| EXECUTION_DESCRIPTION | VARCHAR2(256) | | Optional description of the last execution |
| EXECUTION_START | DATE | | Execution start date and time of the task |
| EXECUTION_END | DATE | | Execution end date and time of the task |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATUS | VARCHAR2(11) | | Current operational status of the task:<br>• INITIAL - Initial state of the task; no recommendations are present<br>• EXECUTING - Task is currently running<br>• INTERRUPTED - Task analysis was interrupted by the user. Recommendation data, if present, can be viewed and reported at this time.<br>• COMPLETED - Task successfully completed the analysis operation. Recommendation data can be viewed and reported.<br>• ERROR - An error occurred during the analysis operation. Recommendations, if present, can be viewed and reported at this time. |
| STATUS_MESSAGE | VARCHAR2(4000) | | Informational message provided by the advisor regarding the status |
| PCT_COMPLETION_TIME | NUMBER | | Percent completion, in terms of time, of the task when it is executing |
| PROGRESS_METRIC | NUMBER | | Metric that measures the progress of the task in terms of quality. Each advisor may have its own metric. |
| METRIC_UNITS | VARCHAR2(64) | | Unit of the metric used to measure progress |
| ACTIVITY_COUNTER | NUMBER | | Counter that is updated frequently by the advisor, denoting that useful work is being performed |
| RECOMMENDATION_COUNT | NUMBER | | Number of recommendations produced |
| ERROR_MESSAGE | VARCHAR2(4000) | | Informational message or an error message indicating the current operation or condition |
| SOURCE | VARCHAR2(128) | | Optional name that identifies the creator of the task |
| HOW_CREATED | VARCHAR2(30) | | Optional task or template on which the object was based |
| READ_ONLY | VARCHAR2(5) | | Indicates whether the task is read-only (TRUE) or not (FALSE) |
| SYSTEM_TASK | VARCHAR2(5) | | Indicates whether the task is a system task (TRUE) or not (FALSE). The automatic SQL tuning task, SYS_AUTO_SQL_TUNING_TASK, is one example of a system task. |
| ADVISOR_ID | NUMBER | NOT NULL | Unique identifier for the advisor |
| STATUS# | NUMBER | | Reserved for internal use |

**See Also:**

"USER_ADVISOR_TASKS"

# DBA_ADVISOR_TEMPLATES

DBA_ADVISOR_TEMPLATES displays information about all templates in the database.

### Related View

USER_ADVISOR_TEMPLATES displays information about the templates owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_ID | NUMBER | NOT NULL | Unique identifier of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| DESCRIPTION | VARCHAR2(256) | | User-supplied description of the task |
| ADVISOR_NAME | VARCHAR2(128) | | Advisor associated with the task |
| CREATED | DATE | NOT NULL | Creation date of the task |
| LAST_MODIFIED | DATE | NOT NULL | Date on which the task was last modified |
| SOURCE | VARCHAR2(128) | | Optional task or template on which the template was based |
| READ_ONLY | VARCHAR2(5) | | Indicates whether the task can be modified or deleted (TRUE) or not (FALSE) |

**See Also:**

"USER_ADVISOR_TEMPLATES"

# DBA_ADVISOR_USAGE

DBA_ADVISOR_USAGE displays the usage information for each type of advisor in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADVISOR_ID | NUMBER | NOT NULL | Type of the advisor |
| ADVISOR_NAME | VARCHAR2(128) | NOT NULL | Name of the advisor |
| LAST_EXEC_TIME | DATE | NOT NULL | Date of the last execution |
| NUM_EXECS | NUMBER | NOT NULL | Cumulative number of executions |
| NUM_DB_REPORTS | NUMBER | NOT NULL | Cumulative number of reports |
| FIRST_REPORT_TIME | DATE | | Time of the first report |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_REPORT_TIME | DATE | | Time of the last report |

# DBA_ALERT_HISTORY

DBA_ALERT_HISTORY describes a time-limited history of alerts which are no longer outstanding.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEQUENCE_ID | NUMBER | NOT NULL | Alert sequence number |
| REASON_ID | NUMBER | | ID of the alert reason |
| OWNER | VARCHAR2(128) | | Owner of the object on which the alert was issued |
| OBJECT_NAME | VARCHAR2(513) | | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject (for example: partition) |
| OBJECT_TYPE | VARCHAR2(64) | | Type of the object (for example: table, tablespace) |
| REASON | VARCHAR2(4000) | | Reason for the alert |
| TIME_SUGGESTED | TIMESTAMP(6) WITH TIME ZONE | | Time when the alert was last updated |
| CREATION_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time when the alert was first created |
| SUGGESTED_ACTION | VARCHAR2(4000) | | Advice of the recommended action |
| ADVISOR_NAME | VARCHAR2(128) | | Name of the advisor to be invoked for more information |
| METRIC_VALUE | NUMBER | | Value of the related metrics |
| MESSAGE_TYPE | VARCHAR2(12) | | Message type:<br>• Notification<br>• Warning |
| MESSAGE_GROUP | VARCHAR2(64) | | Name of the message group to which the alert belongs |
| MESSAGE_LEVEL | NUMBER | | Severity message level (1 to 32) |
| HOSTING_CLIENT_ID | VARCHAR2(64) | | ID of the client or security group to which the alert relates |
| MODULE_ID | VARCHAR2(64) | | ID of the module that originated the alert |
| PROCESS_ID | VARCHAR2(128) | | Process ID |
| HOST_ID | VARCHAR2(256) | | DNS host name of the originating host |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HOST_NW_ADDR | VARCHAR2(256) | | IP or other network address of the originating host |
| INSTANCE_NAME | VARCHAR2(16) | | Originating instance name |
| INSTANCE_NUMBER | NUMBER | | Originating instance number |
| USER_ID | VARCHAR2(128) | | User ID |
| EXECUTION_CONTEXT_ID | VARCHAR2(128) | | Execution Context ID |
| ERROR_INSTANCE_ID | VARCHAR2(142) | | ID of an error instance plus a sequence number |
| RESOLUTION | VARCHAR2(7) | | Resolution:<br>• Cleared<br>• N/A |
| PDB_NAME | VARCHAR2(128) | | PDB name |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_ALERT_HISTORY_DETAIL

DBA_ALERT_HISTORY_DETAIL describes a time-limited history of cleared and outstanding alerts.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEQUENCE_ID | NUMBER | NOT NULL | Alert sequence number |
| REASON_ID | NUMBER | | ID of the alert reason |
| OWNER | VARCHAR2(128) | | Owner of the object on which alert is issued |
| OBJECT_NAME | VARCHAR2(513) | | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject (for example: partition) |
| OBJECT_TYPE | VARCHAR2(64) | | Type of the object (for example: table, tablespace) |
| REASON | VARCHAR2(4000) | | Reason for the alert |
| TIME_SUGGESTED | TIMESTAMP(6) WITH TIME ZONE | | Time when the alert was last updated |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CREATION_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time when the alert was first created |
| SUGGESTED_ACTION | VARCHAR2(4000) | | Advice of the recommended action |
| ADVISOR_NAME | VARCHAR2(128) | | Name of the advisor to be invoked for more information |
| METRIC_VALUE | NUMBER | | Value of the related metrics |
| MESSAGE_TYPE | VARCHAR2(12) | | Message type:<br>• Notification<br>• Warning |
| MESSAGE_GROUP | VARCHAR2(64) | | Name of the message group to which the alert belongs |
| MESSAGE_LEVEL | NUMBER | | Severity level (1 to 32) |
| HOSTING_CLIENT_ID | VARCHAR2(64) | | ID of the client or security group to which the alert relates |
| MODULE_ID | VARCHAR2(64) | | ID of the module that originated the alert |
| PROCESS_ID | VARCHAR2(128) | | Process id |
| HOST_ID | VARCHAR2(256) | | DNS host name of the originating host |
| HOST_NW_ADDR | VARCHAR2(256) | | IP or other network address of originating host |
| INSTANCE_NAME | VARCHAR2(16) | | Originating instance name |
| INSTANCE_NUMBER | NUMBER | | Originating instance number |
| USER_ID | VARCHAR2(128) | | User ID |
| EXECUTION_CONTEXT_ID | VARCHAR2(128) | | Execution Context ID |
| ERROR_INSTANCE_ID | VARCHAR2(142) | | ID of an error instance plus a sequence number |
| RESOLUTION | VARCHAR2(11) | | Resolution:<br>• Cleared<br>• Outstanding<br>• N/A |
| STATE_TRANSITION_NUMBER | NUMBER | NOT NULL | Sequence number of the state transition for the alert |
| PDB_NAME | VARCHAR2(128) | | PDB name |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## DBA_ALL_TABLES

DBA_ALL_TABLES describes all object tables and relational tables in the database. Its columns are the same as those in ALL_ALL_TABLES.

**See Also:**

"ALL_ALL_TABLES"

## DBA_APPLICATION_ROLES

DBA_APPLICATION_ROLES describes all the roles that have authentication policy functions defined.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROLE | VARCHAR2(128) | NOT NULL | Name of the application role |
| SCHEMA | VARCHAR2(128) | NOT NULL | Schema of the authorized package |
| PACKAGE | VARCHAR2(128) | NOT NULL | Name of the authorized package |

## DBA_APPLY

DBA_APPLY displays information about all apply processes in the database. Its columns are the same as those in ALL_APPLY.

**See Also:**

"ALL_APPLY"

## DBA_APPLY_CHANGE_HANDLERS

DBA_APPLY_CHANGE_HANDLERS displays information about the change handlers on all tables in the database. Its columns are the same as those in ALL_APPLY_CHANGE_HANDLERS.

**See Also:**

"ALL_APPLY_CHANGE_HANDLERS"

# DBA_APPLY_CONFLICT_COLUMNS

DBA_APPLY_CONFLICT_COLUMNS displays information about the conflict handlers on all tables in the database. Its columns are the same as those in ALL_APPLY_CONFLICT_COLUMNS.

**See Also:**

"ALL_APPLY_CONFLICT_COLUMNS"

# DBA_APPLY_DML_HANDLERS

DBA_APPLY_DML_HANDLERS displays information about the DML handlers on all tables in the database. Its columns are the same as those in ALL_APPLY_DML_HANDLERS.

**See Also:**

"ALL_APPLY_DML_HANDLERS"

# DBA_APPLY_ENQUEUE

DBA_APPLY_ENQUEUE displays information about the apply enqueue actions for all rules in the database. Its columns are the same as those in ALL_APPLY_ENQUEUE.

**See Also:**

"ALL_APPLY_ENQUEUE"

# DBA_APPLY_ERROR

DBA_APPLY_ERROR displays information about the error transactions generated by all apply processes in the database. Its columns are the same as those in ALL_APPLY_ERROR.

**See Also:**

"ALL_APPLY_ERROR"

# DBA_APPLY_ERROR_MESSAGES

DBA_APPLY_ERROR_MESSAGES displays information about the individual messages in all of the error transactions generated by all apply processes in the database. Its columns are the same as those in ALL_APPLY_ERROR_MESSAGES.

For XStream inbound servers, each message in an error transaction is an LCR.

**Note:**

- Messages that were spilled from memory to hard disk do not appear in this view.
- This view does not contain information related to XStream outbound servers.

**See Also:**

"ALL_APPLY_ERROR_MESSAGES"

## DBA_APPLY_EXECUTE

DBA_APPLY_EXECUTE displays information about the apply execute actions for all rules in the database. Its columns are the same as those in ALL_APPLY_EXECUTE.

**See Also:**

"ALL_APPLY_EXECUTE"

## DBA_APPLY_INSTANTIATED_GLOBAL

DBA_APPLY_INSTANTIATED_GLOBAL displays information about databases for which an instantiation SCN has been set. Its columns are the same as those in ALL_APPLY_INSTANTIATED_GLOBAL.

**See Also:**

"ALL_APPLY_INSTANTIATED_GLOBAL"

## DBA_APPLY_INSTANTIATED_OBJECTS

DBA_APPLY_INSTANTIATED_OBJECTS displays information about objects for which an instantiation SCN has been set. Its columns are the same as those in ALL_APPLY_INSTANTIATED_OBJECTS.

**See Also:**

"ALL_APPLY_INSTANTIATED_OBJECTS"

## DBA_APPLY_INSTANTIATED_SCHEMAS

DBA_APPLY_INSTANTIATED_SCHEMAS displays information about schemas for which an instantiation SCN has been set. Its columns are the same as those in ALL_APPLY_INSTANTIATED_SCHEMAS.

**See Also:**

"ALL_APPLY_INSTANTIATED_SCHEMAS"

## DBA_APPLY_KEY_COLUMNS

DBA_APPLY_KEY_COLUMNS displays information about the substitute key columns for all tables in the database. Its columns are the same as those in ALL_APPLY_KEY_COLUMNS.

**See Also:**

"ALL_APPLY_KEY_COLUMNS"

## DBA_APPLY_OBJECT_DEPENDENCIES

DBA_APPLY_OBJECT_DEPENDENCIES displays information about the object dependencies for all apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| PARENT_OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Parent of the object owner |
| PARENT_OBJECT_NAME | VARCHAR2(128) | NOT NULL | Parent of the named object |

## DBA_APPLY_PARAMETERS

DBA_APPLY_PARAMETERS displays information about the parameters for all apply processes in the database. Its columns are the same as those in ALL_APPLY_PARAMETERS.

**See Also:**

"ALL_APPLY_PARAMETERS"

## DBA_APPLY_PROGRESS

DBA_APPLY_PROGRESS displays information about the progress made by all apply processes in the database. Its columns are the same as those in ALL_APPLY_PROGRESS.

**See Also:**

"ALL_APPLY_PROGRESS"

# DBA_APPLY_SPILL_TXN

DBA_APPLY_SPILL_TXN displays information about the transactions spilled from memory to hard disk by all apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| APPLY_NAME | VARCHAR2(128) | NOT NULL | Name of the apply process that spilled one or more transactions |
| XIDUSN | NUMBER | NOT NULL | Transaction ID undo segment number |
| XIDSLT | NUMBER | NOT NULL | Transaction ID slot number |
| XIDSQN | NUMBER | NOT NULL | Transaction ID sequence number |
| FIRST_SCN | NUMBER | NOT NULL | SCN of the first message in the transaction |
| MESSAGE_COUNT | NUMBER | | Number of messages spilled for the transaction |
| FIRST_MESSAGE_CREATE_TIME | DATE | | Source creation time of the first message in the transaction |
| SPILL_CREATION_TIME | DATE | | Time the first message was spilled |
| FIRST_POSITION | RAW(64) | | Position of the first message in this transaction. This column is populated only for an XStream inbound server. |
| TRANSACTION_ID | VARCHAR2(128) | | Transaction ID of the spilled transaction |

# DBA_APPLY_TABLE_COLUMNS

DBA_APPLY_TABLE_COLUMNS displays, for all tables in the database, information about the nonkey table columns for which Oracle Streams apply processes do not detect conflicts for updates and deletes. Its columns are the same as those in ALL_APPLY_TABLE_COLUMNS.

**See Also:**

"ALL_APPLY_TABLE_COLUMNS"

# DBA_APPLY_VALUE_DEPENDENCIES

DBA_APPLY_VALUE_DEPENDENCIES displays information about the value dependencies for all apply processes in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DEPENDENCY_NAME | VARCHAR2(128) | NOT NULL | Name of the dependency |
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column |
| COLUMN_POSITION | NUMBER | | Position of the column |

# DBA_AQ_AGENT_PRIVS

DBA_AQ_AGENT_PRIVS displays information about the registered AQ agents that are mapped to all users in the database.

### Related View

USER_AQ_AGENT_PRIVS displays information about the registered AQ agents that are mapped to the current user. This view does not display the DB_USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AGENT_NAME | VARCHAR2(128) | NOT NULL | Name of the AQ agent |
| DB_USERNAME | VARCHAR2(128) | | Name of the database user that the agent maps to |
| HTTP_ENABLED | VARCHAR2(4) | | Indicates whether the agent is allowed to access AQ through HTTP (YES) or not (NO) |
| SMTP_ENABLED | VARCHAR2(4) | | Indicates whether the agent is allowed to access AQ through SMTP (YES) or not (NO) |

**See Also:**

"USER_AQ_AGENT_PRIVS"

# DBA_AQ_AGENTS

DBA_AQ_AGENTS displays information about all registered AQ agents in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AGENT_NAME | VARCHAR2(128) | NOT NULL | Name of the AQ agent |
| HTTP_ENABLED | VARCHAR2(4) | | Indicates whether the agent is allowed to access AQ through HTTP (YES) or not (NO) |
| SMTP_ENABLED | VARCHAR2(4) | | Indicates whether the agent is allowed to access AQ through SMTP (YES) or not (NO) |

# DBA_ARGUMENTS

DBA_ARGUMENTS lists the arguments of the functions and procedures that are available in the database. Its columns are the same as those in ALL_ARGUMENTS.

**See Also:**

- "ALL_ARGUMENTS"
- "DBA_PROCEDURES" for information about the functions and procedures that are available in the database

## DBA_ASSEMBLIES

DBA_ASSEMBLIES provides information about all assemblies in the database. Its columns are the same as those in ALL_ASSEMBLIES.

**See Also:**

"ALL_ASSEMBLIES"

## DBA_ASSOCIATIONS

DBA_ASSOCIATIONS describes all user-defined statistics in the database. Its columns are the same as those in ALL_ASSOCIATIONS.

**See Also:**

"ALL_ASSOCIATIONS"

## DBA_ATTRIBUTE_TRANSFORMATIONS

DBA_ATTRIBUTE_TRANSFORMATIONS displays information about the transformation functions for all transformations in the database. Its columns are the same as those in ALL_ATTRIBUTE_TRANSFORMATIONS.

**See Also:**

"ALL_ATTRIBUTE_TRANSFORMATIONS"

## DBA_AUDIT_EXISTS

DBA_AUDIT_EXISTS displays audit trail entries produced by AUDIT EXISTS and AUDIT NOT EXISTS.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OS_USERNAME | VARCHAR2(255) | | Operating system login username of the user whose actions were audited |
| USERNAME | VARCHAR2(128) | | Name (not ID number) of the user whose actions were audited |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| TERMINAL | VARCHAR2(255) | | Identifier of the user's terminal |
| TIMESTAMP | DATE | | Date and time of the creation of the audit trail entry (date and time of user login for entries created by AUDIT SESSION) in the local database session time zone |
| OWNER | VARCHAR2(128) | | Intended creator of the non-existent object |
| OBJ_NAME | VARCHAR2(128) | | Name of the object affected by the action |
| ACTION_NAME | VARCHAR2(28) | | Name of the action type corresponding to the numeric code in the ACTION column in DBA_AUDIT_TRAIL |
| NEW_OWNER | VARCHAR2(128) | | Owner of the object named in the NEW_NAME column |
| NEW_NAME | VARCHAR2(128) | | New name of an object after a RENAME or the name of the underlying object |
| OBJ_PRIVILEGE | VARCHAR2(16) | | Object privileges granted or revoked by a GRANT or REVOKE statement |
| SYS_PRIVILEGE | VARCHAR2(40) | | System privileges granted or revoked by a GRANT or REVOKE statement |
| GRANTEE | VARCHAR2(128) | | Name of the grantee specified in a GRANT or REVOKE statement |
| SESSIONID | NUMBER | NOT NULL | Numeric ID for each Oracle session |
| ENTRYID | NUMBER | NOT NULL | Numeric ID for each audit trail entry in the session |
| STATEMENTID | NUMBER | NOT NULL | Numeric ID for each statement run |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RETURNCODE | NUMBER | NOT NULL | Oracle error code generated by the action. Some useful values:<br>• 0 - Action succeeded<br>• 2004 - Security violation |
| CLIENT_ID | VARCHAR2(128) | | Client identifier in each Oracle session |
| ECONTEXT_ID | VARCHAR2(64) | | Application execution context identifier |
| SESSION_CPU | NUMBER | | Amount of CPU time used by each Oracle session |
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the creation of the audit trail entry (timestamp of user login for entries created by AUDIT SESSION) in UTC (Coordinated Universal Time) time zone |
| PROXY_SESSIONID | NUMBER | | Proxy session serial number, if an enterprise user has logged in through the proxy mechanism |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier for the user, if the user has logged in as an enterprise user |
| INSTANCE_NUMBER | NUMBER | | Instance number as specified by the INSTANCE_NUMBER initialization parameter |
| OS_PROCESS | VARCHAR2(16) | | Operating System process identifier of the Oracle process |
| TRANSACTIONID | RAW(8) | | Transaction identifier of the transaction in which the object is accessed or modified |
| SCN | NUMBER | | System change number (SCN) of the query |
| SQL_BIND | NVARCHAR2(2000) | | Bind variable data of the query |
| SQL_TEXT | NVARCHAR2(2000) | | SQL text of the query |
| OBJ_EDITION_NAME | VARCHAR2(128) | | Name of the edition containing the audited object |

**Note:**

The SQL_BIND and SQL_TEXT columns are only populated if the AUDIT_TRAIL initialization parameter is set to db, extended.

**See Also:**

"AUDIT_TRAIL"

# DBA_AUDIT_MGMT_CLEAN_EVENTS

DBA_AUDIT_MGMT_CLEAN_EVENTS displays information about the history of audit trail cleanup or purge events. Periodically, you should delete the contents of this view so that it will not grow too large.

**Note:**

This view is populated in any Oracle Database where auditing is enabled, regardless of whether pre-Oracle Database 12*c* auditing or unified auditing is enabled for the database.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AUDIT_TRAIL | VARCHAR2(28) | | Audit trail that was cleaned at the time of the event:<br>• STANDARD AUDIT TRAIL<br>• FGA AUDIT TRAIL<br>• STANDARD AND FGA AUDIT TRAIL<br>• OS AUDIT TRAIL<br>• XML AUDIT TRAIL<br>• OS AND XML AUDIT TRAIL<br>• ALL AUDIT TRAILS |
| RAC_INSTANCE | NUMBER | NOT NULL | Instance number indicating the Oracle RAC instance that was cleaned up at the time of the event; 0 implies not applicable |
| CLEANUP_TIME | TIMESTAMP(6) WITH TIME ZONE | | Timestamp when the cleanup event completed |
| DELETE_COUNT | NUMBER | | Number of audit records or audit files that were deleted at the time of the event |
| WAS_FORCED | VARCHAR2(3) | | Indicates whether a forced cleanup occurred (YES) or not (NO); forced cleanup bypasses the last archive timestamp set |

**Note:**

In a read-only database, including an Oracle Active Data Guard physical standby database, this view is not populated. Instead, a summary of cleanup events is written to the respective database instance's alert log file.

# DBA_AUDIT_MGMT_CLEANUP_JOBS

DBA_AUDIT_MGMT_CLEANUP_JOBS displays information about the configured audit trail purge jobs.

**Note:**

This view is populated in any Oracle Database where auditing is enabled, regardless of whether pre-Oracle Database 12*c* auditing or unified auditing is enabled for the database.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_NAME | VARCHAR2(100) | NOT NULL | Name of the audit trail purge job |
| JOB_STATUS | VARCHAR2(8) | | Current status of the audit trail purge job (ENABLED) or (DISABLED) |
| AUDIT_TRAIL | VARCHAR2(28) | | Audit trail for which the audit trail purge job is configured:<br>• STANDARD AUDIT TRAIL<br>• FGA AUDIT TRAIL<br>• STANDARD AND FGA AUDIT TRAIL<br>• OS AUDIT TRAIL<br>• XML AUDIT TRAIL<br>• OS AND XML AUDIT TRAIL<br>• ALL AUDIT TRAILS<br>• UNIFIED AUDIT TRAIL |
| JOB_FREQUENCY | VARCHAR2(100) | | Frequency at which the audit trail purge job runs |
| USE_LAST_ARCHIVE_TIMESTAMP[1] | VARCHAR2(3) | | Indicates whether the audit trail purge job invocation uses the last archive timestamp. The last archive timestamp is shown in the LAST_ARCHIVE_TS column of the DBA_AUDIT_MGMT_LAST_ARCH_TS view. Possible values:<br>• YES - Indicates that the audit trail purge job invocation uses the last archive timestamp<br>• NO - Indicates that the audit trail purge job invocation uses the last archive timestamp |
| JOB_CONTAINER[1] | VARCHAR2(7) | | In a CDB, indicates whether audit trail purge job will be performed only in the current container or in all the containers. Possible values:<br>• CURRENT - Indicates that audit trail purge job will be performed only in the current container<br>• ALL - Indicates that audit trail purge job will be performed in all the containers<br>In a non-CDB, the value in this column is always CURRENT. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DBA_AUDIT_MGMT_LAST_ARCH_TS"

# DBA_AUDIT_MGMT_CONFIG_PARAMS

DBA_AUDIT_MGMT_CONFIG_PARAMS displays information about the currently configured audit trail properties that are used by the DBMS_AUDIT_MGMT PL/SQL package.

**Note:**

This view is populated in any Oracle Database where auditing is enabled, regardless of whether pre-Oracle Database 12*c* auditing or unified auditing is enabled for the database.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETER_NAME | VARCHAR2(1024) | NOT NULL | Name of the property |
| PARAMETER_VALUE | VARCHAR2(4000) | | Value of the property |
| AUDIT_TRAIL | VARCHAR2(28) | | Audit trails for which the property is configured:<br>• STANDARD AUDIT TRAIL<br>• FGA AUDIT TRAIL<br>• STANDARD AND FGA AUDIT TRAIL<br>• OS AUDIT TRAIL<br>• XML AUDIT TRAIL<br>• OS AND XML AUDIT TRAIL<br>• ALL AUDIT TRAILS<br>• UNIFIED AUDIT TRAIL |

**Note:**

In a read-only database, including an Oracle Active Data Guard physical standby database, this view is not populated when the DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL PL/SQL procedure is invoked. If the procedure was used when the database was in read-only mode, use DBMS_AUDIT_MGMT.GET_AUDIT_TRAIL to check the value of the property.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the parameters specified with the `DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY` procedure

# DBA_AUDIT_MGMT_LAST_ARCH_TS

`DBA_AUDIT_MGMT_LAST_ARCH_TS` displays information about the last archive timestamps set for audit trail cleanup or purges.

**Note:**

This view is populated in any Oracle Database where auditing is enabled, regardless of whether pre-Oracle Database 12*c* auditing or unified auditing is enabled for the database.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `AUDIT_TRAIL` | `VARCHAR2(20)` | | Audit trail for which the last archive timestamp applies:<br>• `STANDARD AUDIT TRAIL`<br>• `FGA AUDIT TRAIL`<br>• `OS AUDIT TRAIL`<br>• `XML AUDIT TRAIL`<br>• `UNIFIED AUDIT TRAIL` |
| `RAC_INSTANCE` | `NUMBER` | `NOT NULL` | Oracle RAC instance number for which the last archive timestamp applies; `0` implies not applicable |
| `LAST_ARCHIVE_TS` | `TIMESTAMP(6) WITH TIME ZONE` | | Timestamp of the last audit record or audit file that has been archived |
| `DATABASE_ID` | `NUMBER` | `NOT NULL` | Database ID of the audit records to clean up |
| `CONTAINER_GUID`[1] | `VARCHAR2(33)` | `NOT NULL` | GUID of the container of the audit records to clean up |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

In a read-only database, including an Oracle Active Data Guard physical standby database, this view is not populated when `DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP` is invoked. In such a case, use `DBMS_AUDIT_MGMT.GET_LAST_ARCHIVE_TIMESTAMP` to check for the timestamp, if it was configured for the database instance.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about `DBMS_AUDIT_MGMT` subprograms

# DBA_AUDIT_OBJECT

`DBA_AUDIT_OBJECT` displays audit trail records for all objects in the database.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

### Related View

`USER_AUDIT_OBJECT` displays audit trail records for the objects accessible to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `OS_USERNAME` | `VARCHAR2(255)` | | Operating system login username of the user whose actions were audited |
| `USERNAME` | `VARCHAR2(128)` | | Name (not ID number) of the user whose actions were audited |
| `USERHOST` | `VARCHAR2(128)` | | Client host machine name |
| `TERMINAL` | `VARCHAR2(255)` | | Identifier of the user's terminal |
| `TIMESTAMP` | `DATE` | | Date and time of the creation of the audit trail entry (date and time of user login for entries created by `AUDIT SESSION`) in the local database session time zone |
| `OWNER` | `VARCHAR2(128)` | | Creator of the object affected by the action |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJ_NAME | VARCHAR2(128) | | Name of the object affected by the action |
| ACTION_NAME | VARCHAR2(28) | | Name of the action type corresponding to the numeric code in the ACTION column in DBA_AUDIT_TRAIL |
| NEW_OWNER | VARCHAR2(128) | | Owner of the object named in the NEW_NAME column |
| NEW_NAME | VARCHAR2(128) | | New name of an object after a RENAME or the name of the underlying object |
| SES_ACTIONS | VARCHAR2(19) | | Session summary (a string of 16 characters, one for each action type in the order ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE, REFERENCES, and EXECUTE). Positions 14, 15, and 16 are reserved for future use. The characters are:<br>• - - None<br>• S - Success<br>• F - Failure<br>• B - Both |
| COMMENT_TEXT | VARCHAR2(4000) | | Text comment on the audit trail |
| SESSIONID | NUMBER | NOT NULL | Numeric ID for each Oracle session |
| ENTRYID | NUMBER | NOT NULL | Numeric ID for each audit trail entry in the session |
| STATEMENTID | NUMBER | NOT NULL | Numeric ID for each statement run |
| RETURNCODE | NUMBER | NOT NULL | Oracle error code generated by the action. Some useful values:<br>• 0 - Action succeeded<br>• 2004 - Security violation |
| PRIV_USED | VARCHAR2(40) | | System privilege used to execute the action |
| CLIENT_ID | VARCHAR2(128) | | Client identifier in each Oracle session |
| ECONTEXT_ID | VARCHAR2(64) | | Application execution context identifier |
| SESSION_CPU | NUMBER | | Amount of CPU time used by each Oracle session |
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the creation of the audit trail entry (timestamp of user login for entries created by AUDIT SESSION) in UTC (Coordinated Universal Time) time zone |
| PROXY_SESSIONID | NUMBER | | Proxy session serial number, if an enterprise user has logged in through the proxy mechanism |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier for the user, if the user has logged in as an enterprise user |
| INSTANCE_NUMBER | NUMBER | | Instance number as specified by the INSTANCE_NUMBER initialization parameter |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OS_PROCESS | VARCHAR2(16) | | Operating System process identifier of the Oracle process |
| TRANSACTIONID | RAW(8) | | Transaction identifier of the transaction in which the object is accessed or modified |
| SCN | NUMBER | | System change number (SCN) of the query |
| SQL_BIND | NVARCHAR2(2000) | | Bind variable data of the query |
| SQL_TEXT | NVARCHAR2(2000) | | SQL text of the query |
| OBJ_EDITION_NAME | VARCHAR2(128) | | Name of the edition containing the audited object |

**Note:**

The SQL_BIND and SQL_TEXT columns are only populated if the AUDIT_TRAIL initialization parameter is set to db, extended.

**See Also:**

- "AUDIT_TRAIL"
- "USER_AUDIT_OBJECT"

# DBA_AUDIT_POLICIES

DBA_AUDIT_POLICIES describes all fine-grained auditing policies in the database. Its columns are the same as those in ALL_AUDIT_POLICIES.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

**See Also:**

"ALL_AUDIT_POLICIES"

# DBA_AUDIT_POLICY_COLUMNS

DBA_AUDIT_POLICY_COLUMNS describes all fine-grained auditing policy columns in the database. Its columns are the same as those in ALL_AUDIT_POLICY_COLUMNS.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

**See Also:**

"ALL_AUDIT_POLICY_COLUMNS"

# DBA_AUDIT_SESSION

DBA_AUDIT_SESSION displays all audit trail records concerning CONNECT and DISCONNECT.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

### Related View

USER_AUDIT_SESSION displays the audit trail records concerning connections and disconnections of the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OS_USERNAME | VARCHAR2(255) | | Operating system login username of the user whose actions were audited |
| USERNAME | VARCHAR2(128) | | Name (not ID number) of the user whose actions were audited |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERHOST | VARCHAR2(128) | | Client host machine name |
| TERMINAL | VARCHAR2(255) | | Identifier of the user's terminal |
| TIMESTAMP | DATE | | Date and time of the creation of the audit trail entry (date and time of user login for entries created by AUDIT SESSION) in the local database session time zone |
| ACTION_NAME | VARCHAR2(28) | | Name of the action type corresponding to the numeric code in the ACTION column in DBA_AUDIT_TRAIL |
| LOGOFF_TIME | DATE | | Date and time of user log off |
| LOGOFF_LREAD | NUMBER | | Logical reads for the session |
| LOGOFF_PREAD | NUMBER | | Physical reads for the session |
| LOGOFF_LWRITE | NUMBER | | Logical writes for the session |
| LOGOFF_DLOCK | VARCHAR2(40) | | Deadlocks detected during the session |
| SESSIONID | NUMBER | NOT NULL | Numeric ID for each Oracle session |
| RETURNCODE | NUMBER | NOT NULL | Oracle error code generated by the action. Some useful values:<br>• 0 - Action succeeded<br>• 2004 - Security violation |
| CLIENT_ID | VARCHAR2(128) | | Client identifier in each Oracle session |
| SESSION_CPU | NUMBER | | Amount of CPU time used by each Oracle session |
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the creation of the audit trail entry (timestamp of user login for entries created by AUDIT SESSION) in UTC (Coordinated Universal Time) time zone |
| PROXY_SESSIONID | NUMBER | | Proxy session serial number, if an enterprise user has logged in through the proxy mechanism |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier for the user, if the user has logged in as an enterprise user |
| INSTANCE_NUMBER | NUMBER | | Instance number as specified by the INSTANCE_NUMBER initialization parameter |
| OS_PROCESS | VARCHAR2(16) | | Operating System process identifier of the Oracle process |

**See Also:**

"USER_AUDIT_SESSION"

# DBA_AUDIT_STATEMENT

DBA_AUDIT_STATEMENT displays audit trail records for all GRANT, REVOKE, AUDIT, NOAUDIT, and ALTER SYSTEM statements in the database.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

### Related View

USER_AUDIT_STATEMENT displays audit trail records for the GRANT, REVOKE, AUDIT, NOAUDIT, and ALTER SYSTEM statements issued by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OS_USERNAME | VARCHAR2(255) | | Operating system login username of the user whose actions were audited |
| USERNAME | VARCHAR2(128) | | Name (not ID number) of the user whose actions were audited |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| TERMINAL | VARCHAR2(255) | | Identifier of the user's terminal |
| TIMESTAMP | DATE | | Date and time of the creation of the audit trail entry (date and time of user login for entries created by AUDIT SESSION) in the local database session time zone |
| OWNER | VARCHAR2(128) | | Creator of the object affected by the action |
| OBJ_NAME | VARCHAR2(128) | | Name of the object affected by the action |
| ACTION_NAME | VARCHAR2(28) | | Name of the action type corresponding to the numeric code in the ACTION column in DBA_AUDIT_TRAIL |
| NEW_NAME | VARCHAR2(128) | | New name of an object after a RENAME or the name of the underlying object |
| OBJ_PRIVILEGE | VARCHAR2(16) | | Object privileges granted or revoked by a GRANT or REVOKE statement |
| SYS_PRIVILEGE | VARCHAR2(40) | | System privileges granted or revoked by a GRANT or REVOKE statement |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADMIN_OPTION | VARCHAR2(1) | | Signifies the role or system privilege was granted with the ADMIN option |
| GRANTEE | VARCHAR2(128) | | Name of the grantee specified in a GRANT or REVOKE statement |
| AUDIT_OPTION | VARCHAR2(40) | | Auditing option set with the AUDIT statement |
| SES_ACTIONS | VARCHAR2(19) | | Session summary (a string of 16 characters, one for each action type in the order ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE, REFERENCES, and EXECUTE). Positions 14, 15, and 16 are reserved for future use. The characters are:<br>• - - None<br>• S - Success<br>• F - Failure<br>• B - Both |
| COMMENT_TEXT | VARCHAR2(4000) | | Text comment on the audit trail, inserted by the application |
| SESSIONID | NUMBER | NOT NULL | Numeric ID for each Oracle session |
| ENTRYID | NUMBER | NOT NULL | Numeric ID for each audit trail entry in the session |
| STATEMENTID | NUMBER | NOT NULL | Numeric ID for each statement run |
| RETURNCODE | NUMBER | NOT NULL | Oracle error code generated by the action. Some useful values:<br>• 0 - Action succeeded<br>• 2004 - Security violation |
| PRIV_USED | VARCHAR2(40) | | System privilege used to execute the action |
| CLIENT_ID | VARCHAR2(128) | | Client identifier in each Oracle session |
| ECONTEXT_ID | VARCHAR2(64) | | Application execution context identifier |
| SESSION_CPU | NUMBER | | Amount of CPU time used by each Oracle session |
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the creation of the audit trail entry (timestamp of user login for entries created by AUDIT SESSION) in UTC (Coordinated Universal Time) time zone |
| PROXY_SESSIONID | NUMBER | | Proxy session serial number, if an enterprise user has logged in through the proxy mechanism |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier for the user, if the user has logged in as an enterprise user |
| INSTANCE_NUMBER | NUMBER | | Instance number as specified by the INSTANCE_NUMBER initialization parameter |
| OS_PROCESS | VARCHAR2(16) | | Operating System process identifier of the Oracle process |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRANSACTIONID | RAW(8) | | Transaction identifier of the transaction in which the object is accessed or modified |
| SCN | NUMBER | | System change number (SCN) of the query |
| SQL_BIND | NVARCHAR2(2000) | | Bind variable data of the query |
| SQL_TEXT | NVARCHAR2(2000) | | SQL text of the query |
| OBJ_EDITION_NAME | VARCHAR2(128) | | Name of the edition containing the audited object |

**Note:**

The SQL_BIND and SQL_TEXT columns are only populated if the AUDIT_TRAIL initialization parameter is set to db, extended.

**See Also:**

- "AUDIT_TRAIL"
- "USER_AUDIT_STATEMENT"

# DBA_AUDIT_TRAIL

DBA_AUDIT_TRAIL displays all standard audit trail entries.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

### Related View

USER_AUDIT_TRAIL displays the standard audit trail entries related to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OS_USERNAME | VARCHAR2(255) | | Operating system login username of the user whose actions were audited |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | | Name (not ID number) of the user whose actions were audited |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| TERMINAL | VARCHAR2(255) | | Identifier of the user's terminal |
| TIMESTAMP | DATE | | Date and time of the creation of the audit trail entry (date and time of user login for entries created by AUDIT SESSION) in the local database session time zone |
| OWNER | VARCHAR2(128) | | Creator of the object affected by the action |
| OBJ_NAME | VARCHAR2(128) | | Name of the object affected by the action |
| ACTION | NUMBER | NOT NULL | Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. |
| ACTION_NAME | VARCHAR2(28) | | Name of the action type corresponding to the numeric code in the ACTION column |
| NEW_OWNER | VARCHAR2(128) | | Owner of the object named in the NEW_NAME column |
| NEW_NAME | VARCHAR2(128) | | New name of the object after a RENAME or the name of the underlying object |
| OBJ_PRIVILEGE | VARCHAR2(16) | | Object privileges granted or revoked by a GRANT or REVOKE statement |
| SYS_PRIVILEGE | VARCHAR2(40) | | System privileges granted or revoked by a GRANT or REVOKE statement |
| ADMIN_OPTION | VARCHAR2(1) | | Indicates whether the role or system privilege was granted with the ADMIN option |
| GRANTEE | VARCHAR2(128) | | Name of the grantee specified in a GRANT or REVOKE statement |
| AUDIT_OPTION | VARCHAR2(40) | | Auditing option set with the AUDIT statement |
| SES_ACTIONS | VARCHAR2(19) | | Session summary (a string of 16 characters, one for each action type in the order ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE, REFERENCES, and EXECUTE). Positions 14, 15, and 16 are reserved for future use. The characters are:<br>• - - None<br>• S - Success<br>• F - Failure<br>• B - Both |
| LOGOFF_TIME | DATE | | Date and time of user log off |
| LOGOFF_LREAD | NUMBER | | Logical reads for the session |
| LOGOFF_PREAD | NUMBER | | Physical reads for the session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOGOFF_LWRITE | NUMBER | | Logical writes for the session |
| LOGOFF_DLOCK | VARCHAR2(40) | | Deadlocks detected during the session |
| COMMENT_TEXT | VARCHAR2(4000) | | Text comment on the audit trail entry, providing more information about the statement audited<br>Also indicates how the user was authenticated. The method can be one of the following:<br>• DATABASE - Authentication was done by password<br>• NETWORK - Authentication was done by Oracle Net Services or strong authentication<br>• PROXY - Client was authenticated by another user; the name of the proxy user follows the method type |
| SESSIONID | NUMBER | NOT NULL | Numeric ID for each Oracle session. Each user session gets a unique session ID. |
| ENTRYID | NUMBER | NOT NULL | Numeric ID for each audit trail entry in the session. The entry ID is an index of a session's audit entries that starts at 1 and increases to the number of entries that are written. |
| STATEMENTID | NUMBER | NOT NULL | *n*th statement in the user session. The first SQL statement gets a value of 1 and the value is incremented for each subsequent SQL statement. Note that one SQL statement can create more than one audit trail entry (for example, when more than one object is audited from the same SQL statement), and in this case the statement ID remains the same for that statement and the entry ID increases for each audit trail entry created by the statement. |
| RETURNCODE | NUMBER | NOT NULL | Oracle error code generated by the action. Some useful values:<br>• 0 - Action succeeded<br>• 2004 - Security violation |
| PRIV_USED | VARCHAR2(40) | | System privilege used to execute the action |
| CLIENT_ID | VARCHAR2(128) | | Client identifier in each Oracle session |
| ECONTEXT_ID | VARCHAR2(64) | | Application execution context identifier |
| SESSION_CPU | NUMBER | | Amount of CPU time used by each Oracle session |
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the creation of the audit trail entry (timestamp of user login for entries created by AUDIT SESSION) in UTC (Coordinated Universal Time) time zone |
| PROXY_SESSIONID | NUMBER | | Proxy session serial number, if an enterprise user has logged in through the proxy mechanism |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier for the user, if the user has logged in as an enterprise user |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INSTANCE_NUMBER | NUMBER | | Instance number as specified by the INSTANCE_NUMBER initialization parameter |
| OS_PROCESS | VARCHAR2(16) | | Operating System process identifier of the Oracle process |
| TRANSACTIONID | RAW(8) | | Transaction identifier of the transaction in which the object is accessed or modified |
| SCN | NUMBER | | System change number (SCN) of the query |
| SQL_BIND | NVARCHAR2(2000) | | Bind variable data of the query |
| SQL_TEXT | NVARCHAR2(2000) | | SQL text of the query |
| OBJ_EDITION_NAME | VARCHAR2(128) | | Name of the edition containing the audited object |
| DBID | NUMBER | | Database identifier of the audited database |

**Note:**

The SQL_BIND and SQL_TEXT columns are only populated if the AUDIT_TRAIL initialization parameter is set to db, extended.

**See Also:**

- "AUDIT_TRAIL"
- "USER_AUDIT_TRAIL"

# DBA_AUTO_SEGADV_CTL

DBA_AUTO_SEGADV_CTL exposes the control information used by the segment advisor. This information gives the DBA an idea of what is happening in the auto advisor.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AUTO_TASKID | NUMBER | | Unique task ID generated by the auto advisor |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the segment |
| SEGMENT_OWNER | VARCHAR2(128) | | Owner of the segment |
| SEGMENT_NAME | VARCHAR2(128) | | Name of the segment |
| SEGMENT_TYPE | VARCHAR2(18) | | The type of segment (TABLE, INDEX, and so on) |
| PARTITION_NAME | VARCHAR2(128) | | Name of the subsegment (if partitioned) |
| STATUS | VARCHAR2(40) | | Status of the analysis: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NEW - the segment/tablespace has not been analyzed<br>• BEING-PROCESSED - the segment/ tablespace is being processed<br>• COMPLETE - the segment/tablespace has been analyzed<br>• ERROR - problem with the segment/ tablespace |
| REASON | VARCHAR2(40) | | Reason why this segment was chosen |
| REASON_VALUE | NUMBER | | A value associated with the reason |
| CREATION_TIME | TIMESTAMP(6) | | Time when this entry was created |
| PROCESSED_TASKID | NUMBER | | Auto advisor task that was used to process the segment/tablespace |
| END_TIME | TIMESTAMP(6) | | Time at which the advisor task was completed |

# DBA_AUTO_SEGADV_SUMMARY

DBA_AUTO_SEGADV_SUMMARY provides a summary of the auto advisor task runs.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AUTO_TASKID | NUMBER | NOT NULL | Unique auto task ID |
| SNAPID | NUMBER | | Maximum AWR snapid used to process the segments |
| SEGMENTS_SELECTED | NUMBER | | Number of segments chosen for analysis |
| SEGMENTS_PROCESSED | NUMBER | | Number of segments actually processed |
| TABLESPACE_SELECTED | NUMBER | | Number of tablespaces chosen for analysis |
| TABLESPACE_PROCESSED | NUMBER | | Number of tablespaces actually processed |
| RECOMMENDATIONS_COUNT | NUMBER | | Number of recommendations generated |
| START_TIME | TIMESTAMP(6) | | Time at which the auto task was started |
| END_TIME | TIMESTAMP(6) | | Time at which the auto task ended |

# DBA_AUTOTASK_CLIENT

DBA_AUTOTASK_CLIENT displays statistical data for each automated maintenance task over 7-day and 30-day periods.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLIENT_NAME | VARCHAR2(64) | | Name of the client |
| STATUS | VARCHAR2(8) | | Job status: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • ENABLED<br>• DISABLED |
| CONSUMER_GROUP | VARCHAR2(128) | | Consumer group used for normal priority jobs for this client |
| CLIENT_TAG | VARCHAR2(2) | | Tag used to identify jobs for this client |
| PRIORITY_OVERRIDE | VARCHAR2(7) | | User-specified priority at which the task executes:<br>• URGENT<br>• HIGH<br>• MEDIUM<br>• LOW |
| ATTRIBUTES | VARCHAR2(4000) | | Attributes of the client |
| WINDOW_GROUP | VARCHAR2(64) | | Window group used to schedule jobs |
| SERVICE_NAME | VARCHAR2(64) | | Name of the service on which jobs will execute for the client |
| RESOURCE_PERCENTAGE | NUMBER | | Percentage of maintenance resources for high priority maintenance tasks for this client |
| USE_RESOURCE_ESTIMATES | VARCHAR2(5) | | Indicates whether resource estimates are used for this client (TRUE) or not (FALSE) |
| MEAN_JOB_DURATION | INTERVAL DAY(9) TO SECOND(9) | | Average elapsed time for a job for this client (in seconds) |
| MEAN_JOB_CPU | INTERVAL DAY(9) TO SECOND(9) | | Average CPU time for a job submitted by this client (in seconds) |
| MEAN_JOB_ATTEMPTS | NUMBER | | Average number of attempts it takes to complete a task |
| MEAN_INCOMING_TASKS_7_DAYS | NUMBER | | Average number of incoming tasks at the Maintenance Window Start over the last 7 days |
| MEAN_INCOMING_TASKS_30_DAYS | NUMBER | | Average number of incoming tasks at the Maintenance Window Start over the last 30 days |
| TOTAL_CPU_LAST_7_DAYS | INTERVAL DAY(9) TO SECOND(9) | | Cumulative CPU time used by the jobs for this client over the last 7 days (in seconds) |
| TOTAL_CPU_LAST_30_DAYS | INTERVAL DAY(9) TO SECOND(9) | | Cumulative CPU time used by the jobs for this client over the last 30 days (in seconds) |
| MAX_DURATION_LAST_7_DAYS | INTERVAL DAY(3) TO SECOND(0) | | Maximum elapsed time for a job over the last 7 days (in seconds) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MAX_DURATION_LAST_30_DAYS | INTERVAL DAY(3) TO SECOND(0) | | Maximum elapsed time for a job over the last 30 days (in seconds) |
| WINDOW_DURATION_LAST_7_DAYS | INTERVAL DAY(9) TO SECOND(9) | | Total time during which the client was active during the last 7 days |
| WINDOW_DURATION_LAST_30_DAYS | INTERVAL DAY(9) TO SECOND(9) | | Total time during which the client was active during the last 30 days |
| LAST_CHANGE | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of last configuration change for the client |

# DBA_AUTOTASK_CLIENT_HISTORY

DBA_AUTOTASK_CLIENT_HISTORY displays per-window history of job execution counts for each automated maintenance task. This information is viewable in the Job History page of Enterprise Manager.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLIENT_NAME | VARCHAR2(64) | | Name of the client |
| WINDOW_NAME | VARCHAR2(261) | | Name of the maintenance window |
| WINDOW_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Maintenance window start time |
| WINDOW_DURATION | INTERVAL DAY(9) TO SECOND(6) | | Window duration (NULL for currently open window) |
| JOBS_CREATED | NUMBER | | Number of jobs created on behalf of the client in this window |
| JOBS_STARTED | NUMBER | | Number of jobs started on behalf of the client during the maintenance window |
| JOBS_COMPLETED | NUMBER | | Number of jobs successfully completed on behalf of the client during the maintenance window |
| WINDOW_END_TIME | TIMESTAMP(6) WITH TIME ZONE | | Window end time |

# DBA_AUTOTASK_CLIENT_JOB

DBA_AUTOTASK_CLIENT_JOB displays information about currently running Scheduler jobs created for automated maintenance tasks. It provides information about some objects targeted by those jobs, as well as some additional statistics from previous instantiations of the same task. Some of this additional data is taken from generic Scheduler views.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLIENT_NAME | VARCHAR2(64) | | Name of the client |
| JOB_NAME | VARCHAR2(65) | | Name of the job |
| JOB_SCHEDULER_STATUS | VARCHAR2(15) | | Job status:<br>• DISABLED<br>• RETRY SCHEDULED<br>• SCHEDULED<br>• RUNNING<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• REMOTE<br>• SUCCEEDED<br>• CHAIN_STALLED |
| TASK_NAME | VARCHAR2(64) | | Name of the task being performed |
| TASK_TARGET_TYPE | VARCHAR2(64) | | Type of the target being processed |
| TASK_TARGET_NAME | VARCHAR2(513) | NOT NULL | Name of the target |
| TASK_PRIORITY | VARCHAR2(7) | | Priority of the task:<br>• URGENT<br>• HIGH<br>• MEDIUM<br>• LOW |
| TASK_OPERATION | VARCHAR2(64) | | Operation performed on the object |

# DBA_AUTOTASK_JOB_HISTORY

DBA_AUTOTASK_JOB_HISTORY displays the history of automated maintenance task job runs. Jobs are added to this view after they finish executing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLIENT_NAME | VARCHAR2(64) | | Name of the automated maintenance client |
| WINDOW_NAME | VARCHAR2(261) | | Name of the maintenance window |
| WINDOW_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Start time of the maintenance window |
| WINDOW_DURATION | INTERVAL DAY(9) TO SECOND(6) | | Duration of the maintenance window |
| JOB_NAME | VARCHAR2(261) | | Name of the maintenance job |
| JOB_STATUS | VARCHAR2(30) | | Status of the maintenance job |
| JOB_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Start time of the maintenance job |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_DURATION | INTERVAL DAY(3) TO SECOND(0) | | Duration of the maintenance job |
| JOB_ERROR | NUMBER | | Error code for the job (if any) |
| JOB_INFO | VARCHAR2(4000) | | Additional information about the job |

# DBA_AUTOTASK_OPERATION

DBA_AUTOTASK_OPERATION displays all automated maintenance task operations for each client.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLIENT_NAME | VARCHAR2(64) | | Name of the client |
| OPERATION_NAME | VARCHAR2(64) | | Name of the operation |
| OPERATION_TAG | VARCHAR2(3) | | Tag for the operation |
| PRIORITY_OVERRIDE | VARCHAR2(7) | | User-specified priority at which the task executes:<br>• URGENT<br>• HIGH<br>• MEDIUM<br>• LOW |
| ATTRIBUTES | VARCHAR2(4000) | | Attributes of the operation |
| USE_RESOURCE_ESTIMATES | VARCHAR2(5) | | Indicates whether resource usage estimates are used for the operation (TRUE) or not (FALSE) |
| STATUS | VARCHAR2(8) | | Job status:<br>• ENABLED<br>• DISABLED |
| LAST_CHANGE | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the last change |

# DBA_AUTOTASK_SCHEDULE

DBA_AUTOTASK_SCHEDULE displays the schedule of maintenance windows for the next 32 days for each client.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WINDOW_NAME | VARCHAR2(128) | | Name of the maintenance window |
| START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Projected start time of the window |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DURATION | INTERVAL DAY(3) TO SECOND(0) | | Currently defined duration of the window (NULL for the currently open window) |

# DBA_AUTOTASK_STATUS

DBA_AUTOTASK_STATUS displays status information for automated maintenance.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATUS | VARCHAR2(8) | | Shows the status of automated maintenance. Possible values are:<br>• ENABLED<br>• DISABLED<br>• ALLOWED<br>• INVALID |
| LAST_CHANGE | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of last status change |

# DBA_AUTOTASK_TASK

DBA_AUTOTASK_TASK displays information about current and past automated maintenance tasks.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLIENT_NAME | VARCHAR2(64) | | Name of the client |
| TASK_NAME | VARCHAR2(64) | | Name of the task being performed |
| TASK_TARGET_TYPE | VARCHAR2(64) | | Target type of the task |
| TASK_TARGET_NAME | VARCHAR2(513) | NOT NULL | Name of the target |
| OPERATION_NAME | VARCHAR2(64) | | Operation performed on the object |
| ATTRIBUTES | VARCHAR2(4000) | | Attributes of the task |
| TASK_PRIORITY | NUMBER | | Task priority, relative to other tasks for this Client |
| PRIORITY_OVERRIDE | NUMBER | | Task priority as overridden by the user |
| STATUS | VARCHAR2(8) | | Status of the task:<br>• DISABLED<br>• DEFERRED<br>• ENABLED |
| DEFERRED_WINDOW_NAME | VARCHAR2(65) | | Appropriate window for this task |
| CURRENT_JOB_NAME | VARCHAR2(65) | | Name of the currently scheduled job, if any |
| JOB_SCHEDULER_STATUS | VARCHAR2(15) | | Job status:<br>• DISABLED |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • RETRY SCHEDULED<br>• SCHEDULED<br>• RUNNING<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• REMOTE<br>• SUCCEEDED<br>• CHAIN_STALLED |
| ESTIMATE_TYPE | VARCHAR2(7) | | Type of resource estimates applied:<br>• DERIVED<br>• FORCED<br>• LOCKED |
| ESTIMATED_WEIGHT | NUMBER | | Task weight indicator |
| ESTIMATED_DURATION | NUMBER | | Estimated elapsed time for the job (in seconds) |
| ESTIMATED_CPU_TIME | NUMBER | | Estimated CPU time for the job (in seconds) |
| ESTIMATED_TEMP | NUMBER | | Estimated temporary space usage for the job (in KB) |
| ESTIMATED_DOP | NUMBER | | Estimated degree of parallelism for the job |
| ESTIMATED_IO_RATE | NUMBER | | Estimated I/O utilization for the job (in KB per second) |
| ESTIMATED_UNDO_RATE | NUMBER | | Estimated UNDO generation rate for the job (in KB per second) |
| RETRY_COUNT | NUMBER | | Number of attempts to perform this task since the last successful attempt |
| LAST_GOOD_DATE | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the last successful attempt to perform this task |
| LAST_GOOD_PRIORITY | NUMBER | | Job priority of the last successful attempt to perform this task |
| LAST_GOOD_DURATION | NUMBER | | Elapsed time (in seconds) of the last successful attempt to perform this task |
| LAST_GOOD_CPU_TIME | NUMBER | | CPU time for the job (in seconds) of the last successful attempt to perform this task |
| LAST_GOOD_TEMP | NUMBER | | Temporary space usage for the job (in KB) of the last successful attempt to perform this task |
| LAST_GOOD_DOP | NUMBER | | Peak degree of parallelism for the job during the last successful attempt to perform this task |
| LAST_GOOD_IO_RATE | NUMBER | | I/O utilization rate for the job (in KB per second) of the last successful attempt to perform this task |
| LAST_GOOD_UNDO_RATE | NUMBER | | NDO generation rate (in KB per second) of the last successful attempt to perform this task |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_GOOD_CPU_WAIT | NUMBER | | Resource Manager wait time (in seconds) of the last successful attempt to perform this task |
| LAST_GOOD_IO_WAIT | NUMBER | | Resource Manager wait time (in seconds) of the last successful attempt to perform this task |
| LAST_GOOD_UNDO_WAIT | NUMBER | | Resource Manager wait time (in seconds) of the last successful attempt to perform this task |
| LAST_GOOD_TEMP_WAIT | NUMBER | | Resource Manager wait time (in seconds) of the last successful attempt to perform this task |
| LAST_GOOD_CONCURRENCY | NUMBER | | Concurrency wait time (in seconds) of the last successful attempt to perform this task |
| LAST_GOOD_CONTENTION | NUMBER | | Contention wait time (in seconds) of the last successful attempt to perform this task |
| NEXT_TRY_DATE | TIMESTAMP(6) WITH TIME ZONE | | Next projected start time for the deferred maintenance window |
| LAST_TRY_DATE | TIMESTAMP(6) WITH TIME ZONE | | Time at which the task was last attempted |
| LAST_TRY_PRIORITY | NUMBER | | Task priority at the time of the last attempt |
| LAST_TRY_RESULT | VARCHAR2(36) | | Result code of the last execution of the task:<br>• SUCCEEDED<br>• FAILED<br>• STOPPED BY USER ACTION<br>• STOPPED AT END OF MAINTENANCE WINDOW<br>• STOPPED AT INSTANCE SHUTDOWN<br>• STOPPED |
| LAST_TRY_DURATION | NUMBER | | Elapsed time of the last run (in seconds) |
| LAST_TRY_CPU_TIME | NUMBER | | CPU time during the last run (in seconds) |
| LAST_TRY_TEMP | NUMBER | | Temporary space usage for the job (in KB) for the last run |
| LAST_TRY_DOP | NUMBER | | Peak degree of parallelism for the job during the last run |
| LAST_TRY_IO_RATE | NUMBER | | I/O rate during the last run (in seconds) |
| LAST_TRY_UNDO_RATE | NUMBER | | UNDO generation rate during the last run (in seconds) |
| LAST_TRY_CPU_WAIT | NUMBER | | Time spent waiting for CPU during the last run (in seconds) |
| LAST_TRY_IO_WAIT | NUMBER | | Time spent waiting for I/O during the last run (in seconds) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_TRY_UNDO_WAIT | NUMBER | | Time spent waiting for UNDO during the last run (in seconds) |
| LAST_TRY_TEMP_WAIT | NUMBER | | Time spent waiting for temporary space during the last run (in seconds) |
| LAST_TRY_CONCURRENCY | NUMBER | | Concurrency wait time during the last run (in seconds) |
| LAST_TRY_CONTENTION | NUMBER | | Contention wait time during the last run (in seconds) |
| MEAN_GOOD_DURATION | NUMBER | | Average elapsed time for successful executions of this task (in seconds) |
| MEAN_GOOD_CPU_TIME | NUMBER | | Average CPU time for successful executions of this task (in seconds) |
| MEAN_GOOD_TEMP | NUMBER | | Average temporary space usage for successful executions of this task (in KB) |
| MEAN_GOOD_DOP | NUMBER | | Average peak degree of parallelism for successful executions of this task |
| MEAN_GOOD_IO | NUMBER | | Average I/O utilization for successful executions of this task (in KB per second) |
| MEAN_GOOD_UNDO | NUMBER | | Average UNDO generation rate for this task (in KB per second) |
| MEAN_GOOD_CPU_WAIT | NUMBER | | Average time waiting for CPU for successful executions of this task (in seconds) |
| MEAN_GOOD_IO_WAIT | NUMBER | | Average time waiting for I/O for successful executions of this task (in seconds) |
| MEAN_GOOD_UNDO_WAIT | NUMBER | | Average time waiting for UNDO for successful executions of this task (in seconds) |
| MEAN_GOOD_TEMP_WAIT | NUMBER | | Average time waiting for temporary space for successful executions of this task (in seconds) |
| MEAN_GOOD_CONCURRENCY | NUMBER | | Average concurrency wait time for successful executions of this task (in seconds) |
| MEAN_GOOD_CONTENTION | NUMBER | | Average contention wait time for successful executions of this task (in seconds) |
| INFO_FIELD_1 | VARCHAR2(4000) | | Client-interpreted information |
| INFO_FIELD_2 | CLOB | | Client-interpreted information |
| INFO_FIELD_3 | NUMBER | | Client-interpreted information |
| INFO_FIELD_4 | NUMBER | | Client-interpreted information |

# DBA_AUTOTASK_WINDOW_CLIENTS

DBA_AUTOTASK_WINDOW_CLIENTS displays the windows that belong to MAINTENANCE_WINDOW_GROUP, along with the Enabled or Disabled status for the window for each maintenance task. Primarily used by Enterprise Manager.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WINDOW_NAME | VARCHAR2(128) | NOT NULL | Name of the maintenance window |
| WINDOW_NEXT_TIME | TIMESTAMP(6) WITH TIME ZONE | | Next scheduled window open time unless the window is disabled |
| WINDOW_ACTIVE | VARCHAR2(5) | | Indicates whether the window is currently active (open) (TRUE) or not (FALSE) |
| AUTOTASK_STATUS | VARCHAR2(8) | | Status of the automated maintenance task subsystem:<br>• ENABLED<br>• DISABLED |
| OPTIMIZER_STATS | VARCHAR2(8) | | Status of optimizer statistics gathering:<br>• ENABLED<br>• DISABLED |
| SEGMENT_ADVISOR | VARCHAR2(8) | | Status of Segment Advisor:<br>• ENABLED<br>• DISABLED |
| SQL_TUNE_ADVISOR | VARCHAR2(8) | | Status of SQL Tuning Advisor:<br>• ENABLED<br>• DISABLED |
| HEALTH_MONITOR | VARCHAR2(8) | | Status of Health Monitor:<br>• ENABLED<br>• DISABLED |

# DBA_AUTOTASK_WINDOW_HISTORY

DBA_AUTOTASK_WINDOW_HISTORY displays historical information for automated maintenance task windows.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WINDOW_NAME | VARCHAR2(261) | | Name of the maintenance window |
| WINDOW_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Window start time |
| WINDOW_END_TIME | TIMESTAMP(6) WITH TIME ZONE | | Window end time |

# DBA_AW_PS

DBA_AW_PS describes the page spaces in all analytic workspaces in the database. Its columns are the same as those in ALL_AW_PS.

**See Also:**

"ALL_AW_PS"

# DBA_AWS

DBA_AWS describes all analytic workspaces in the database. Its columns are the same as those in ALL_AWS.

**See Also:**

"ALL_AWS"

# DBA_BASE_TABLE_MVIEWS

DBA_BASE_TABLE_MVIEWS describes all materialized views using materialized view logs in the database. Its columns are the same as those in ALL_BASE_TABLE_MVIEWS.

**See Also:**

"ALL_BASE_TABLE_MVIEWS"

# DBA_BLOCKERS

DBA_BLOCKERS displays a session if it is not waiting for a locked object but is holding a lock on an object for which another session is waiting. In an Oracle RAC environment, this only applies if the blocker is on the same instance.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HOLDING_SESSION | NUMBER | | Session holding a lock |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_CAPTURE

DBA_CAPTURE displays information about all capture processes in the database. Its columns are the same as those in ALL_CAPTURE.

**See Also:**

"ALL_CAPTURE"

# DBA_CAPTURE_EXTRA_ATTRIBUTES

DBA_CAPTURE_EXTRA_ATTRIBUTES displays information about the extra attributes for all capture processes in the database. Its columns are the same as those in ALL_CAPTURE_EXTRA_ATTRIBUTES.

**See Also:**

"ALL_CAPTURE_EXTRA_ATTRIBUTES"

# DBA_CAPTURE_PARAMETERS

DBA_CAPTURE_PARAMETERS displays information about the parameters for all capture processes in the database. Its columns are the same as those in ALL_CAPTURE_PARAMETERS.

**See Also:**

"ALL_CAPTURE_PARAMETERS"

# DBA_CAPTURE_PREPARED_DATABASE

DBA_CAPTURE_PREPARED_DATABASE displays information about when the local database was prepared for instantiation. Its columns are the same as those in ALL_CAPTURE_PREPARED_DATABASE.

**See Also:**

- "ALL_CAPTURE_PREPARED_DATABASE"
- *Oracle Streams Concepts and Administration* for more information about the possible values for the supplemental logging columns in this view

# DBA_CAPTURE_PREPARED_SCHEMAS

DBA_CAPTURE_PREPARED_SCHEMAS displays information about all schemas prepared for instantiation at the local database. Its columns are the same as those in ALL_CAPTURE_PREPARED_SCHEMAS.

**See Also:**

- "ALL_CAPTURE_PREPARED_SCHEMAS"
- *Oracle Streams Concepts and Administration* for more information about the possible values for the supplemental logging columns in this view

## DBA_CAPTURE_PREPARED_TABLES

DBA_CAPTURE_PREPARED_TABLES displays information about all tables prepared for instantiation at the local database. Its columns are the same as those in ALL_CAPTURE_PREPARED_TABLES.

**See Also:**

- "ALL_CAPTURE_PREPARED_TABLES"
- *Oracle Streams Concepts and Administration* for more information about the possible values for the supplemental logging columns in this view

## DBA_CATALOG

DBA_CATALOG lists all tables, views, clusters, synonyms, and sequences in the database. Its columns are the same as those in "ALL_CATALOG".

## DBA_CDB_RSRC_PLAN_DIRECTIVES

DBA_CDB_RSRC_PLAN_DIRECTIVES provides information about all the CDB resource plan directives.

**Note:**

ORA$DEFAULT_PDB_DIRECTIVE is the default directive for PDBs. For more information about ORA$DEFAULT_PDB_DIRECTIVE, see *Oracle Database Administrator's Guide*.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PLAN | VARCHAR2(128) | | Name of the CDB resource plan to which this directive belongs |
| PLUGGABLE_DATABASE | VARCHAR2(128) | | Name of the PDB referred to. NULL for profile directives |
| PROFILE | VARCHAR2(128) | | For internal use only |
| DIRECTIVE_TYPE | VARCHAR2(30) | | For internal use only |
| SHARES | NUMBER | | Resource allocation, expressed in shares |
| UTILIZATION_LIMIT | NUMBER | | Maximum resource utilization allowed, expressed in percentage |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARALLEL_SERVER_LIMIT | NUMBER | | Maximum percentage of the parallel target used before queueing subsequent parallel queries |
| MEMORY_MIN | NUMBER | | For internal use only |
| MEMORY_LIMIT | NUMBER | | For internal use only |
| COMMENTS | VARCHAR2(2000) | | Text comment on the resource plan directive |
| STATUS | VARCHAR2(128) | | PENDING if it is part of the pending area, NULL otherwise |
| MANDATORY | VARCHAR2(3) | | Whether the resource plan directive is mandatory. Mandatory plans cannot be deleted. |

# DBA_CDB_RSRC_PLANS

DBA_CDB_RSRC_PLANS provides information about all the CDB resource plans.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PLAN_ID | NUMBER | NOT NULL | CDB resource plan ID |
| PLAN | VARCHAR2(128) | | CDB resource plan name |
| COMMENTS | VARCHAR2(2000) | | Text comment on the CDB resource plan |
| STATUS | VARCHAR2(128) | | PENDING if it is part of the pending area, NULL otherwise |
| MANDATORY | VARCHAR2(3) | | Whether the resource plan is mandatory. Mandatory plans cannot be deleted. |

# DBA_CHANGE_NOTIFICATION_REGS

DBA_CHANGE_NOTIFICATION_REGS describes all change notification registrations in the database.

### Related View

USER_CHANGE_NOTIFICATION_REGS describes the change notification registrations owned by the current user. This view does not display the USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(31) | | Owner of the registration |
| REGID | NUMBER | | Internal registration ID |
| REGFLAGS | NUMBER | | Registration flags |
| CALLBACK | VARCHAR2(256) | | Notification callback |
| OPERATIONS_FILTER | NUMBER | | Operations filter (if specified) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CHANGELAG | NUMBER | | Transaction lag between notifications (if specified) |
| TIMEOUT | NUMBER | | Registration timeout (if specified) |
| TABLE_NAME | VARCHAR2(63) | | Name of the registered table |

**See Also:**

"USER_CHANGE_NOTIFICATION_REGS"

# DBA_CLU_COLUMNS

DBA_CLU_COLUMNS maps all table columns to related cluster columns.

### Related View

USER_CLU_COLUMNS maps all table columns owned by the current user to related cluster columns. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the cluster |
| CLUSTER_NAME | VARCHAR2(128) | NOT NULL | Name of the cluster |
| CLU_COLUMN_NAME | VARCHAR2(128) | NOT NULL | Key column in the cluster |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Clustered table name |
| TAB_COLUMN_NAME | VARCHAR2(4000) | | Key column or attribute of the object type column |

**See Also:**

"USER_CLU_COLUMNS"

# DBA_CLUSTER_HASH_EXPRESSIONS

DBA_CLUSTER_HASH_EXPRESSIONS lists hash functions for all hash clusters in the database. Its columns are the same as those in "ALL_CLUSTER_HASH_EXPRESSIONS".

# DBA_CLUSTERING_DIMENSIONS

DBA_CLUSTERING_DIMENSIONS describes dimension tables associated with all tables with an attribute clustering clause in the database. Its columns are the same as those in ALL_CLUSTERING_DIMENSIONS.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_DIMENSIONS"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## DBA_CLUSTERING_JOINS

`DBA_CLUSTERING_JOINS` describes joins to the dimension tables associated with all tables with an attribute clustering clause in the database. Its columns are the same as those in `ALL_CLUSTERING_JOINS`.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_JOINS"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## DBA_CLUSTERING_KEYS

`DBA_CLUSTERING_KEYS` describes clustering keys for all tables with an attribute clustering clause. Its columns are the same as those in `ALL_CLUSTERING_KEYS`.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_KEYS"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## DBA_CLUSTERING_TABLES

`DBA_CLUSTERING_TABLES` describes all the tables with an attribute clustering clause. Its columns are the same as those in `ALL_CLUSTERING_TABLES`.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_TABLES"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## DBA_CLUSTERS

DBA_CLUSTERS describes all clusters in the database. Its columns are the same as those in "ALL_CLUSTERS".

## DBA_CODE_ROLE_PRIVS

DBA_CODE_ROLE_PRIVS describes all the roles that are associated with program units in the database. Its columns are the same as those in ALL_CODE_ROLE_PRIVS.

**See Also:**

"ALL_CODE_ROLE_PRIVS"

## DBA_COL_COMMENTS

DBA_COL_COMMENTS displays comments on the columns of all tables and views in the database. Its columns are the same as those in ALL_COL_COMMENTS.

**See Also:**

"ALL_COL_COMMENTS"

## DBA_COL_PENDING_STATS

DBA_COL_PENDING_STATS describes the pending statistics of all columns in the database. Its columns are the same as those in ALL_COL_PENDING_STATS.

**See Also:**

"ALL_COL_PENDING_STATS"

## DBA_COL_PRIVS

DBA_COL_PRIVS describes all column object grants in the database.

**Related View**

USER_COL_PRIVS describes the column object grants for which the current user is the object owner, grantor, or grantee.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the column |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |

**See Also:**

"USER_COL_PRIVS"

# DBA_COLL_TYPES

DBA_COLL_TYPES describes all named collection types (arrays, nested tables, object tables, and so on) in the database. Its columns are the same as those in "ALL_COLL_TYPES".

# DBA_COMMON_AUDIT_TRAIL

DBA_COMMON_AUDIT_TRAIL displays all standard and fine-grained audit trail entries, mandatory and SYS audit records written in XML format.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AUDIT_TYPE | VARCHAR2(22) | | Audit trail type:<br>• Standard Audit<br>• Standard XML Audit<br>• Fine Grained Audit<br>• Fine Grained XML Audit<br>• SYS XML Audit<br>• Mandatory XML Audit |
| SESSION_ID | NUMBER | | Numeric ID for the Oracle session |
| PROXY_SESSIONID | NUMBER | | Proxy session serial number, if an enterprise user has logged in through the proxy mechanism |
| STATEMENTID | NUMBER | | Numeric ID for the statement run; a statement may cause multiple audit records |
| ENTRYID | NUMBER | | Numeric ID for the audit trail entry in the session |
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the audited operation (timestamp of user login for entries created by AUDIT SESSION) in the session's time zone |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier for the user, if the user has logged in as an enterprise user |
| DB_USER | VARCHAR2(128) | | Database user name of the user whose actions were audited |
| CLIENT_ID | VARCHAR2(128) | | Client identifier in the Oracle session |
| ECONTEXT_ID | VARCHAR2(64) | | Application execution context identifier |
| EXT_NAME | VARCHAR2(4000) | | User external name |
| OS_USER | VARCHAR2(255) | | Operating system login user name of the user whose actions were audited |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| OS_PROCESS | VARCHAR2(16) | | Operating system process identifier of the Oracle process |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TERMINAL | VARCHAR2(255) | | Identifier of the user's terminal |
| INSTANCE_NUMBER | NUMBER | | Instance number as specified by the INSTANCE_NUMBER initialization parameter |
| OBJECT_SCHEMA | VARCHAR2(128) | | Owner of the audited object |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object affected by the action |
| POLICY_NAME | VARCHAR2(128) | | Name of the Fine-Grained Auditing Policy |
| NEW_OWNER | VARCHAR2(128) | | Owner of the object named in the NEW_NAME column |
| NEW_NAME | VARCHAR2(128) | | New name of the object after a RENAME or the name of the underlying object |
| ACTION | NUMBER | | Numeric action type code. The corresponding name of the action type is in the STATEMENT_TYPE column. |
| STATEMENT_TYPE | VARCHAR2(28) | | Name of the action type corresponding to the numeric code in the ACTION column |
| AUDIT_OPTION | VARCHAR2(40) | | Auditing option set with the AUDIT statement |
| TRANSACTIONID | RAW(8) | | Transaction identifier of the transaction in which the object was accessed or modified |
| RETURNCODE | NUMBER | | Oracle error code generated by the action (0 if the action succeeded) |
| SCN | NUMBER | | System change number (SCN) of the query |
| COMMENT_TEXT | VARCHAR2(4000) | | Text comment on the audit trail entry, providing more information about the statement audited<br>Also indicates how the user was authenticated:<br>• DATABASE - Authentication was done by password<br>• NETWORK - Authentication was done by Oracle Net Services or the Advanced Networking Option<br>• PROXY - Client was authenticated by another user. The name of the proxy user follows the method type. |
| SQL_BIND | NVARCHAR2(2000) | | Bind variable data of the query |
| SQL_TEXT | NVARCHAR2(2000) | | SQL text of the query |
| OBJ_PRIVILEGE | VARCHAR2(16) | | Object privileges granted or revoked by a GRANT or REVOKE statement |
| SYS_PRIVILEGE | VARCHAR2(40) | | System privileges granted or revoked by a GRANT or REVOKE statement |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADMIN_OPTION | VARCHAR2(1) | | Indicates whether the role or system privilege was granted with the ADMIN option |
| OS_PRIVILEGE | VARCHAR2(7) | | Operating privilege (SYSDBA or SYSOPER), if any, used in the session. If no privilege is used, it will be NONE. |
| GRANTEE | VARCHAR2(128) | | Name of the grantee specified in a GRANT or REVOKE statement |
| PRIV_USED | VARCHAR2(40) | | System privilege used to execute the action |
| SES_ACTIONS | VARCHAR2(19) | | Session summary (a string of 16 characters, one for each action type in the order ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE, REFERENCES, and EXECUTE). Positions 14, 15, and 16 are reserved for future use. The characters are:<br>• - - None<br>• S - Success<br>• F - Failure<br>• B - Both |
| LOGOFF_TIME | DATE | | Timestamp of user log off |
| LOGOFF_LREAD | NUMBER | | Number of logical reads in the session |
| LOGOFF_PREAD | NUMBER | | Number of physical reads in the session |
| LOGOFF_LWRITE | NUMBER | | Number of logical writes for the session |
| LOGOFF_DLOCK | VARCHAR2(40) | | Number of deadlocks detected during the session |
| SESSION_CPU | NUMBER | | Amount of CPU time used by the Oracle session |
| OBJ_EDITION_NAME | VARCHAR2(128) | | Name of the edition containing the audited object |
| DBID | NUMBER | | Database identifier of the audited database |

**Note:**

The SQL_BIND and SQL_TEXT columns are only populated if the AUDIT_TRAIL initialization parameter is set to db, extended or xml, extended or if the AUDIT_SYS_OPERATIONS initialization parameter is set to TRUE.

**See Also:**

- "AUDIT_SYS_OPERATIONS"
- "AUDIT_TRAIL"

# DBA_COMPARISON

DBA_COMPARISON displays information about all comparison objects in the database.

### Related View

USER_COMPARISON displays information about the comparison objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the comparison |
| COMPARISON_NAME | VARCHAR2(128) | NOT NULL | Name of the comparison |
| COMPARISON_MODE | VARCHAR2(5) | | Mode of the comparison:<br>• TABLE |
| SCHEMA_NAME | VARCHAR2(128) | | Schema name of the local object |
| OBJECT_NAME | VARCHAR2(128) | | Name of the local object |
| OBJECT_TYPE | VARCHAR2(17) | | Type of the local object:<br>• TABLE<br>• VIEW<br>• SYNONYM<br>• MATERIALIZED VIEW |
| REMOTE_SCHEMA_NAME | VARCHAR2(128) | | Schema name of the remote object |
| REMOTE_OBJECT_NAME | VARCHAR2(128) | | Name of the remote object |
| REMOTE_OBJECT_TYPE | VARCHAR2(17) | | Type of the remote object:<br>• TABLE<br>• VIEW<br>• SYNONYM<br>• MATERIALIZED VIEW |
| DBLINK_NAME | VARCHAR2(128) | | Database link name to the remote database |
| SCAN_MODE | VARCHAR2(9) | | Scan mode of the comparison: FULL<br>• FULL<br>• RANDOM<br>• CYCLIC<br>• CUSTOM |
| SCAN_PERCENT | NUMBER | | Scan percent of the comparison; applicable to random and cyclic modes |
| CYCLIC_INDEX_VALUE | VARCHAR2(4000) | | Last index column value used in a cyclic scan |
| NULL_VALUE | VARCHAR2(4000) | | Value to use for NULL columns |
| LOCAL_CONVERGE_TAG | RAW(2000) | | Local Streams tag used while performing converge DMLs |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REMOTE_CONVERGE_TAG | RAW(2000) | | Remote Streams tag used while performing converge DMLs |
| MAX_NUM_BUCKETS | NUMBER | | Suggested maximum number of buckets in a scan |
| MIN_ROWS_IN_BUCKET | NUMBER | | Suggested minimum number of rows in a bucket |
| LAST_UPDATE_TIME | TIMESTAMP(6) | | Time that this row was last updated |

**See Also:**

"USER_COMPARISON"

# DBA_COMPARISON_COLUMNS

DBA_COMPARISON_COLUMNS displays information about the columns for all comparison objects in the database.

### Related View

USER_COMPARISON_COLUMNS displays information about the columns for the comparison objects owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the comparison |
| COMPARISON_NAME | VARCHAR2(128) | NOT NULL | Name of the comparison |
| COLUMN_POSITION | NUMBER | NOT NULL | Position of the column |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column |
| INDEX_COLUMN | VARCHAR2(1) | | Indicates whether the column is an index column (Y) or not (N) |

**See Also:**

"USER_COMPARISON_COLUMNS"

# DBA_COMPARISON_ROW_DIF

DBA_COMPARISON_ROW_DIF displays information about the differing rows in all comparison scans in the database.

### Related View

USER_COMPARISON_ROW_DIF displays information about the differing rows in the comparison scans owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the comparison |
| COMPARISON_NAME | VARCHAR2(128) | NOT NULL | Name of the comparison |
| SCAN_ID | NUMBER | NOT NULL | Scan ID for the comparison scan |
| LOCAL_ROWID | ROWID | | Local rowid of the differing row |
| REMOTE_ROWID | ROWID | | Remote rowid of the differing row |
| INDEX_VALUE | VARCHAR2(4000) | | Index column value of the differing row |
| STATUS | VARCHAR2(3) | | Status of the differing row:<br>• SUC<br>• DIF |
| LAST_UPDATE_TIME | TIMESTAMP(6) | | Time that this row was last updated |

**See Also:**

"USER_COMPARISON_ROW_DIF"

# DBA_COMPARISON_SCAN

DBA_COMPARISON_SCAN displays information about all comparison scans in the database.

### Related View

USER_COMPARISON_SCAN displays information about the comparison scans owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the comparison scan |
| COMPARISON_NAME | VARCHAR2(128) | NOT NULL | Name of the comparison scan |
| SCAN_ID | NUMBER | NOT NULL | Scan ID |
| PARENT_SCAN_ID | NUMBER | | Scan ID of the immediate parent scan |
| ROOT_SCAN_ID | NUMBER | | Scan ID of the root (top-most) parent |
| STATUS | VARCHAR2(16) | | Status of the scan:<br>• SUC<br>• BUCKET DIF<br>• FINAL BUCKET DIF<br>• ROW DIF |
| CURRENT_DIF_COUNT | NUMBER | | Current cumulative (including children) diff count of the scan |
| INITIAL_DIF_COUNT | NUMBER | | Initial cumulative (including children) diff count of the scan |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COUNT_ROWS | NUMBER | | Number of rows in the scan |
| SCAN_NULLS | VARCHAR2(1) | | Indicates whether NULLs are part of this scan (Y) or not (N) |
| LAST_UPDATE_TIME | TIMESTAMP(6) | | Time that this row was last updated |

**See Also:**

"USER_COMPARISON_SCAN"

# DBA_COMPARISON_SCAN_VALUES

DBA_COMPARISON_SCAN_VALUES displays information about the values for all comparison scans in the database.

### Related View

USER_COMPARISON_SCAN_VALUES displays information about the values for the comparison scans owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the comparison scan |
| COMPARISON_SCAN | VARCHAR2(128) | NOT NULL | Name of the comparison scan |
| SCAN_ID | NUMBER | NOT NULL | Scan ID |
| COLUMN_POSITION | NUMBER | NOT NULL | Column position, as in DBA_COMPARISON_COLUMNS |
| MIN_VALUE | VARCHAR2(4000) | | Minimum value of the scan |
| MAX_VALUE | VARCHAR2(4000) | | Maximum value of the scan |
| LAST_UPDATE_TIME | TIMESTAMP(6) | | Time that this row was last updated |

**See Also:**

"USER_COMPARISON_SCAN_VALUES"

# DBA_CONNECT_ROLE_GRANTEES

DBA_CONNECT_ROLE_GRANTEES displays information about users who are granted the CONNECT privilege.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | User or schema to which the CONNECT role is granted |
| PATH_OF_CONNECT_ROLE_GRANT | VARCHAR2(4000) | | The path of role inheritance through which the grantee is granted the CONNECT role |
| ADMIN_OPT | VARCHAR2(3) | | Whether or not the grantee was granted the ADMIN option for the CONNECT role |

# DBA_CONS_COLUMNS

DBA_CONS_COLUMNS describes all columns in the database that are specified in constraints. Its columns are the same as those in ALL_CONS_COLUMNS.

**See Also:**

"ALL_CONS_COLUMNS"

# DBA_CONS_OBJ_COLUMNS

DBA_CONS_OBJ_COLUMNS displays information about the types that object columns (or attributes) or collection elements have been constrained to, in all tables in the database. Its columns are the same as those in ALL_CONS_OBJ_COLUMNS.

**See Also:**

"ALL_CONS_OBJ_COLUMNS"

# DBA_CONSTRAINTS

DBA_CONSTRAINTS describes all constraint definitions on all tables in the database. Its columns are the same as those in ALL_CONSTRAINTS.

**See Also:**

"ALL_CONSTRAINTS"

# DBA_CONTAINER_DATA

DBA_CONTAINER_DATA displays default (user-level) and object-specific CONTAINER_DATA attributes for container data objects.

Objects created with the CONTAINER_DATA clause include CONTAINER_DATA attributes.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | | Name of the user whose attribute is described by this row |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DEFAULT_ATTR | CHAR2(1) | | An indicator of whether the attribute is a default attribute |
| OWNER | VARCHAR2(128) | | Name of the object owner if the attribute is object-specific |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object if the attribute is object-specific |
| ALL_CONTAINERS | VARCHAR2(1) | | An indicator of whether this attribute applies to all containers |
| CONTAINER_NAME | VARCHAR2(128) | | Name of a container included in this attribute if it does not apply to all containers |

**See Also:**

For more information about container data objects:

- "CDB_* Views"
- "ALL_TABLES"
- "ALL_VIEWS"
- "ALL_VIEWS_AE"
- "V$ Views"
- "GV$ Views"
- *Oracle Database Administrator's Guide*
- *Oracle Database Security Guide*

# DBA_CONTEXT

DBA_CONTEXT provides all context namespace information in the database. Its columns are the same as those in "ALL_CONTEXT".

# DBA_CPOOL_INFO

DBA_CPOOL_INFO displays configuration information about all Database Resident Connection Pools in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CONNECTION_POOL | VARCHAR2(128) | | Name of the connection pool |
| STATUS | VARCHAR2(16) | | Status of the pool:<br>• ACTIVE<br>• INACTIVE |
| MINSIZE | NUMBER | | Minimum number of pooled servers that are always alive in the pool |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MAXSIZE | NUMBER | | Maximum number of pooled servers in the pool |
| INCRSIZE | NUMBER | | Number of pooled servers by which the pool is incremented if servers are unavailable when a client application request is received |
| SESSION_CACHED_CURSORS | NUMBER | | Number of session cursors to cache in each pooled server |
| INACTIVITY_TIMEOUT | NUMBER | | Maximum time (in seconds) that the pooled server can stay idle in the pool. After this time, the server is terminated. |
| MAX_THINK_TIME | NUMBER | | Maximum time of inactivity (in seconds) for a client after obtaining a pooled server from the pool. After obtaining a pooled server from the pool, if the client application does not issue a database call for the time specified by this column, then the pooled server is freed and the client connection is terminated. |
| MAX_USE_SESSION | NUMBER | | Number of times a pooled server can be taken and released to the pool |
| MAX_LIFETIME_SESSION | NUMBER | | Time (in seconds) for a pooled server to live in the pool |
| NUM_CBROK | NUMBER | | Number of connection brokers spawned per instance |
| MAXCONN_CBROK | NUMBER | | Maximum number of connections per connection broker |

**See Also:**

- *Oracle Database Administrator's Guide* for more information about the configuration parameters for Database Resident Connection Pooling
- *Oracle Database PL/SQL Packages and Types Reference* for more information on the DBMS_CONNECTION_POOL package

# DBA_CPU_USAGE_STATISTICS

DBA_CPU_USAGE_STATISTICS displays database CPU usage statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| VERSION | VARCHAR2(17) | NOT NULL | Database version |
| TIMESTAMP | DATE | NOT NULL | Time at which the CPU usage changed |
| CPU_COUNT | NUMBER | | CPU count of the database |
| CPU_CORE_COUNT | NUMBER | | CPU core count of the database |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CPU_SOCKET_COUNT | NUMBER | | CPU socket count of the database |

# DBA_CQ_NOTIFICATION_QUERIES

DBA_CQ_NOTIFICATION_QUERIES describes the registered queries for all CQ notifications in the database.

### Related View

USER_CQ_NOTIFICATION_QUERIES describes the registered queries for the CQ notifications owned by the current user. This view does not display the USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| QUERYID | NUMBER | | ID of the query |
| QUERYTEXT | CLOB | | Text of the query |
| REGID | NUMBER | | Registration ID that the query is registered with |
| USERNAME | VARCHAR2(31) | | Name of the user who registered the query |

**See Also:**

"USER_CQ_NOTIFICATION_QUERIES"

# DBA_CREDENTIALS

DBA_CREDENTIALS lists all credentials in the database. Its columns are the same as those in ALL_CREDENTIALS.

**See Also:**

"ALL_CREDENTIALS"

# DBA_CUBE_ATTR_VISIBILITY

DBA_CUBE_ATTR_VISIBILITY describes all OLAP attributes visible for the dimensions, hierarchies, and levels in the database. Its columns are the same as those in ALL_CUBE_ATTR_VISIBILITY.

**See Also:**

"ALL_CUBE_ATTR_VISIBILITY"

# DBA_CUBE_ATTRIBUTES

DBA_CUBE_ATTRIBUTES describes the attributes for all OLAP cube dimensions in the database. Its columns are the same as those in ALL_CUBE_ATTRIBUTES.

**See Also:**

"ALL_CUBE_ATTRIBUTES"

## DBA_CUBE_BUILD_PROCESSES

DBA_CUBE_BUILD_PROCESSES describes all OLAP build processes and maintenance scripts in the database. Its columns are the same as those in ALL_CUBE_BUILD_PROCESSES.

**See Also:**

"ALL_CUBE_BUILD_PROCESSES"

## DBA_CUBE_CALCULATED_MEMBERS

DBA_CUBE_CALCULATED_MEMBERS describes the calculated members for all OLAP cube dimensions in the database. Its columns are the same as those in ALL_CUBE_CALCULATED_MEMBERS.

**See Also:**

"ALL_CUBE_CALCULATED_MEMBERS"

## DBA_CUBE_DIM_LEVELS

DBA_CUBE_DIM_LEVELS describes all OLAP dimension levels in the database. Its columns are the same as those in ALL_CUBE_DIM_LEVELS.

**See Also:**

"ALL_CUBE_DIM_LEVELS"

## DBA_CUBE_DIM_MODELS

DBA_CUBE_DIM_MODELS describes the models for all OLAP dimensions in the database. Its columns are the same as those in ALL_CUBE_DIM_MODELS.

**See Also:**

"ALL_CUBE_DIM_MODELS"

## DBA_CUBE_DIM_VIEW_COLUMNS

DBA_CUBE_DIM_VIEW_COLUMNS describes the columns of the relational views of all OLAP cube dimensions in the database. Its columns are the same as those in ALL_CUBE_DIM_VIEW_COLUMNS.

**See Also:**

"ALL_CUBE_DIM_VIEW_COLUMNS"

## DBA_CUBE_DIM_VIEWS

DBA_CUBE_DIM_VIEWS describes the relational views of all OLAP dimensions in the database. Its columns are the same as those in ALL_CUBE_DIM_VIEWS.

**See Also:**

"ALL_CUBE_DIM_VIEWS"

## DBA_CUBE_DIMENSIONALITY

DBA_CUBE_DIMENSIONALITY describes the dimension order for all OLAP cubes in the database. Its columns are the same as those in ALL_CUBE_DIMENSIONALITY.

**See Also:**

"ALL_CUBE_DIMENSIONALITY"

## DBA_CUBE_DIMENSIONS

DBA_CUBE_DIMENSIONS describes all OLAP cube dimensions in the database. Its columns are the same as those in ALL_CUBE_DIMENSIONS.

**See Also:**

"ALL_CUBE_DIMENSIONS"

## DBA_CUBE_HIER_LEVELS

DBA_CUBE_HIER_LEVELS describes the hierarchy levels for all OLAP cube dimensions in the database. Its columns are the same as those in ALL_CUBE_HIER_LEVELS.

**See Also:**

"ALL_CUBE_HIER_LEVELS"

## DBA_CUBE_HIER_VIEW_COLUMNS

DBA_CUBE_HIER_VIEW_COLUMNS describes the columns of the relational hierarchy views of all OLAP cube dimensions in the database. Its columns are the same as those in ALL_CUBE_HIER_VIEW_COLUMNS.

**See Also:**

"ALL_CUBE_HIER_VIEW_COLUMNS"

## DBA_CUBE_HIER_VIEWS

DBA_CUBE_HIER_VIEWS describes the hierarchies for all OLAP cube dimensions in the database. Its columns are the same as those in ALL_CUBE_HIER_VIEWS.

**See Also:**

"ALL_CUBE_HIER_VIEWS"

## DBA_CUBE_HIERARCHIES

DBA_CUBE_HIERARCHIES describes all OLAP dimension hierarchies in the database. Its columns are the same as those in ALL_CUBE_HIERARCHIES.

**See Also:**

"ALL_CUBE_HIERARCHIES"

## DBA_CUBE_MEASURES

DBA_CUBE_MEASURES describes the measures for all OLAP cubes in the database. Its columns are the same as those in ALL_CUBE_MEASURES.

**See Also:**

"ALL_CUBE_MEASURES"

## DBA_CUBE_NAMED_BUILD_SPECS

DBA_CUBE_NAMED_BUILD_SPECS describes the OLAP cube named build specifications in the database. Its columns are the same as those in ALL_CUBE_NAMED_BUILD_SPECS.

**See Also:**

"ALL_CUBE_NAMED_BUILD_SPECS"

## DBA_CUBE_SUB_PARTITION_LEVELS

DBA_CUBE_SUB_PARTITION_LEVELS describes the OLAP secondary partition levels in the database. Its columns are the same as those in ALL_CUBE_SUB_PARTITION_LEVELS.

**See Also:**

"ALL_CUBE_SUB_PARTITION_LEVELS"

## DBA_CUBE_VIEW_COLUMNS

DBA_CUBE_VIEW_COLUMNS describes the columns of relational views of all OLAP cubes in the database. Its columns are the same as those in ALL_CUBE_VIEW_COLUMNS.

**See Also:**

"ALL_CUBE_VIEW_COLUMNS"

## DBA_CUBE_VIEWS

DBA_CUBE_VIEWS describes the relational views of all OLAP cubes in the database. Its columns are the same as those in ALL_CUBE_VIEWS.

**See Also:**

"ALL_CUBE_VIEWS"

## DBA_CUBES

DBA_CUBES describes all OLAP cubes in the database. Its columns are the same as those in ALL_CUBES.

**See Also:**

"ALL_CUBES"

## DBA_DATA_FILES

DBA_DATA_FILES describes database files.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_NAME | VARCHAR2(513) | | Name of the database file |
| FILE_ID | NUMBER | | File identifier number of the database file |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace to which the file belongs |
| BYTES | NUMBER | | Size of the file in bytes |
| BLOCKS | NUMBER | | Size of the file in Oracle blocks |
| STATUS | VARCHAR2(9) | | File status: AVAILABLE or INVALID (INVALID means that the file number is not in use, for example, a file in a tablespace that was dropped) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RELATIVE_FNO | NUMBER | | Relative file number |
| AUTOEXTENSIBLE | VARCHAR2(3) | | Autoextensible indicator |
| MAXBYTES | NUMBER | | Maximum file size in bytes |
| MAXBLOCKS | NUMBER | | Maximum file size in blocks |
| INCREMENT_BY | NUMBER | | Number of Oracle blocks used as autoextension increment |
| USER_BYTES | NUMBER | | The size of the file available for user data. The actual size of the file minus the USER_BYTES value is used to store file related metadata. |
| USER_BLOCKS | NUMBER | | Number of blocks which can be used by the data |
| ONLINE_STATUS | VARCHAR2(7) | | Online status of the file:<br>• SYSOFF<br>• SYSTEM<br>• OFFLINE<br>• ONLINE<br>• RECOVER |

# DBA_DATAPUMP_JOBS

DBA_DATAPUMP_JOBS identifies all active Data Pump jobs in the database, regardless of their state, on an instance (or on all instances for Real Application Clusters). It also show all Data Pump master tables not currently associated with an active job.

### Related View

USER_DATAPUMP_JOBS displays the Data Pump jobs owned by the current user. This view does not display the OWNER_NAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER_NAME | VARCHAR2(128) | | User that initiated the job |
| JOB_NAME | VARCHAR2(128) | | User-supplied name for the job (or the default name generated by the server) |
| OPERATION | VARCHAR2(30) | | Type of job |
| JOB_MODE | VARCHAR2(30) | | Mode of job |
| STATE | VARCHAR2(30) | | Current job state |
| DEGREE | NUMBER | | Number of worker processes performing the operation |
| ATTACHED_SESSIONS | NUMBER | | Number of sessions attached to the job |
| DATAPUMP_SESSIONS | NUMBER | | Number of Data Pump sessions participating in the job |

**See Also:**

"USER_DATAPUMP_JOBS"

## DBA_DATAPUMP_SESSIONS

DBA_DATAPUMP_SESSIONS identifies the user sessions that are attached to a Data Pump job. The information in this view is useful for determining why a stopped Data Pump operation has not gone away.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER_NAME | VARCHAR2(30) | | User that initiated the job |
| JOB_NAME | VARCHAR2(30) | | User-supplied name for the job (or the default name generated by the server) |
| INSTANCE_ID | NUMBER | | Instance ID |
| SADDR | RAW(4 \| 8) | | Address of the session attached to the job. Can be used with V$SESSION view. |
| SESSION_TYPE | VARCHAR2(14) | | Data Pump session type:<br>• DBMS_DATAPUMP - Data Pump interface process (one for each active instantiation of DBMS_DATAPUMP.OPEN and DBMS_DATAPUMP.ATTACH per job.)<br>• MASTER - master control process (one per job)<br>• WORKER - worker process (1 to *n* per job, depending on degree of parallelism)<br>• EXTERNAL_TABLE - external table data access process (1 to *n*, depending on degree of parallelism, for jobs that use external tables as the data access method for some tables)<br>• OTHER |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_DATAPUMP package

## DBA_DB_LINKS

DBA_DB_LINKS describes all database links in the database. Its columns (except for PASSWORD) are the same as those in ALL_DB_LINKS.

**See Also:**

"ALL_DB_LINKS"

## DBA_DBFS_HS

DBA_DBFS_HS shows all Database File System (DBFS) hierarchical stores.

### Related View

USER_DBFS_HS shows all Database File System hierarchical stores owned by the current user. This view does not display the STOREOWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STORENAME | VARCHAR2(256) | | Name of store |
| STOREOWNER | VARCHAR2(64) | | Owner of store |

**See Also:**

"USER_DBFS_HS"

# DBA_DBFS_HS_COMMANDS

DBA_DBFS_HS_COMMANDS shows all the registered store commands for all Database File System (DBFS) hierarchical stores.

### Related View

USER_DBFS_HS_COMMANDS shows all the registered store commands for all DBFS hierarchical stores owned by current user. This view does not display the STOREOWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STORENAME | VARCHAR2(256) | | Name of store |
| STOREOWNER | VARCHAR2(64) | | Owner of store |
| STORECOMMAND | VARCHAR2(512) | | Store command |
| STOREFLAGS | NUMBER | | Valid values are:<br>• 1 - Indicates that the command is sent to the device before put<br>• 2 - Indicates that the command is sent to the device before get |

**See Also:**

"USER_DBFS_HS_COMMANDS"

# DBA_DBFS_HS_FIXED_PROPERTIES

DBA_DBFS_HS_FIXED_PROPERTIES shows non-modifiable properties of all Database File System (DBFS) hierarchical stores.

### Related View

USER_DBFS_HS_FIXED_PROPERTIES shows non-modifiable properties of all DBFS hierarchical stores owned by current user. This view does not display the STORE_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STORE_NAME | VARCHAR2(128) | NOT NULL | Name of store |
| STORE_OWNER | VARCHAR2(128) | NOT NULL | Owner of store |
| PROP_NAME | VARCHAR2(256) | NOT NULL | Property name |
| PROP_VALUE | VARCHAR2(256) | NOT NULL | Property value |

**See Also:**

"USER_DBFS_HS_FIXED_PROPERTIES"

# DBA_DBFS_HS_PROPERTIES

DBA_DBFS_HS_PROPERTIES shows modifiable properties of all Database File System (DBFS) hierarchical stores.

### Related View

USER_DBFS_HS_PROPERTIES shows modifiable properties of all DBFS hierarchical stores owned by current user. This view does not display the STOREOWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STORENAME | VARCHAR2(256) | | Name of store |
| STOREOWNER | VARCHAR2(64) | | Owner of store |
| PROPERTYNAME | VARCHAR2(256) | | Property name |
| PROPERTYVALUE | VARCHAR2(256) | | Property value |

**See Also:**

"USER_DBFS_HS_PROPERTIES"

# DBA_DDL_LOCKS

DBA_DDL_LOCKS lists all DDL locks held in the database and all outstanding requests for a DDL lock.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SESSION_ID | NUMBER | | Session identifier |
| OWNER | VARCHAR2(30) | | Owner of the lock |
| NAME | VARCHAR2(30) | | Name of the lock |
| TYPE | VARCHAR2(40) | | Lock type:<br>• Cursor<br>• Table/Procedure/Type<br>• Body |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • Trigger<br>• Index<br>• Cluster<br>• Java Source<br>• Java Resource<br>• Java Data |
| MODE_HELD | VARCHAR2(9) | | Lock mode:<br>• None<br>• Null<br>• Share<br>• Exclusive |
| MODE_REQUESTED | VARCHAR2(9) | | Lock request type:<br>• None<br>• Null<br>• Share<br>• Exclusive |

**See Also:**

*Oracle Database Concepts* for more information about DDL locks

## DBA_DEPENDENCIES

DBA_DEPENDENCIES describes all dependencies in the database between procedures, packages, functions, package bodies, and triggers, including dependencies on views created without any database links. Its columns are the same as those in "ALL_DEPENDENCIES".

## DBA_DIGEST_VERIFIERS

DBA_DIGEST_VERIFIERS enables the database administrator to check which users have Digest verifiers stored on disk and the type of hashing algorithm used for the verifiers.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | | Name of the user |
| HAS_DIGEST_VERIFIERS | VARCHAR2(3) | | YES if a Digest verifier exists, NO otherwise |
| DIGEST_TYPE | CHAR(3) | | The type of hashing algorithm used for the Digest verifier. For instance, MD5 for users with MD5 Digest verifiers. If no Digest verifier exists, this column is NULL. |

## DBA_DIM_ATTRIBUTES

DBA_DIM_ATTRIBUTES describes the relationships between dimension levels and functionally dependent columns in the database. The level columns and the

dependent column must be in the same table. This view's columns are the same as those in "ALL_DIM_ATTRIBUTES".

## DBA_DIM_CHILD_OF

DBA_DIM_CHILD_OF describes a hierarchical relationship of 1 to *n* between all the pairs of levels in the dimensions in the database. Its columns are the same as those in "ALL_DIM_CHILD_OF".

## DBA_DIM_HIERARCHIES

DBA_DIM_HIERARCHIES describes all the dimension hierarchies in the database. Its columns are the same as those in "ALL_DIM_HIERARCHIES".

## DBA_DIM_JOIN_KEY

DBA_DIM_JOIN_KEY describes all joins in the database between two dimension tables. The join is always specified between a parent dimension level column and a child column. This view's columns are the same as those in "ALL_DIM_JOIN_KEY".

## DBA_DIM_LEVEL_KEY

DBA_DIM_LEVEL_KEY describes the columns of all dimension levels in the database. This view's columns are the same as those in "ALL_DIM_LEVEL_KEY".

## DBA_DIM_LEVELS

DBA_DIM_LEVELS describes all dimension levels in the database. All columns of a dimension level must come from the same relation. This view's columns are the same as those in "ALL_DIM_LEVELS".

## DBA_DIMENSIONS

DBA_DIMENSIONS represents dimension objects. Its columns are the same as those in "ALL_DIMENSIONS".

## DBA_DIRECTORIES

DBA_DIRECTORIES describes all directory objects in the database. Its columns are the same as those in "ALL_DIRECTORIES".

## DBA_DISCOVERY_SOURCE

DBA_DISCOVERY_SOURCE describes sensitive data discovery import information.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_NAME | VARCHAR2(128) | | The name of the discovery source. In the case of Application Data Model (ADM), this is the ADM instance name. |
| SOURCE_TYPE | VARCHAR2(6) | | The type of the source:<br>• ADM: import from ADM<br>• CUSTOM: custom import<br>• DB: discovered within the database |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CTIME | TIMESTAMP(6) | | The last time sensitive data was imported from this source |

**iSee Also:**

*Oracle Database Security Guide* for more information about transparent sensitive data protection

# DBA_DML_LOCKS

DBA_DML_LOCKS lists all DML locks held in the database and all outstanding requests for a DML lock.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SESSION_ID | NUMBER | | Session holding or acquiring the lock |
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the lock |
| NAME | VARCHAR2(128) | NOT NULL | Name of the lock |
| MODE_HELD | VARCHAR2(13) | | The type of lock held. The values are:<br>• ROWS_S (SS): row share lock<br>• ROW-X (SX): row exclusive lock<br>• SHARE (S): share lock<br>• S/ROW-X (SSX): exclusive lock<br>• NONE: lock requested but not yet obtained |
| MODE_REQUESTED | VARCHAR2(13) | | Lock request type. The values are:<br>• ROWS_S (SS): row share lock<br>• ROW-X (SX): row exclusive lock<br>• SHARE (S): share lock<br>• S/ROW-X (SSX): exclusive lock<br>• NONE: Lock identifier obtained; lock not held or requested |
| LAST_CONVERT | NUMBER | | Time since current mode was granted |
| BLOCKING_OTHERS | VARCHAR2(40) | | Blocking others |

**See Also:**

*Oracle Database Concepts* for more information about lock modes for table locks

# DBA_DMT_FREE_SPACE

DBA_DMT_FREE_SPACE describes the free extents in all dictionary managed tablespaces in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_ID | NUMBER | NOT NULL | Identifier number of the tablespace containing the extent |
| FILE_ID | NUMBER | NOT NULL | File identifier number of the file containing the extent |
| BLOCK_ID | NUMBER | NOT NULL | Starting block number of the extent |
| BLOCKS | NUMBER | NOT NULL | Size of the extent (in Oracle blocks) |

# DBA_DMT_USED_EXTENTS

DBA_DMT_USED_EXTENTS describes the extents comprising the segments in all dictionary managed tablespaces in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEGMENT_FILEID | NUMBER | NOT NULL | File number of the segment header of the extent |
| SEGMENT_BLOCK | NUMBER | NOT NULL | Block number of the segment header of the extent |
| TABLESPACE_ID | NUMBER | NOT NULL | Identifier number of the tablespace containing the extent |
| EXTENT_ID | NUMBER | NOT NULL | Extent number in the segment |
| FILEID | NUMBER | NOT NULL | File identifier number of the file containing the extent |
| BLOCK | NUMBER | NOT NULL | Starting block number of the extent |
| LENGTH | NUMBER | NOT NULL | Number of blocks in the extent |

# DBA_EDITION_COMMENTS

DBA_EDITION_COMMENTS describes the comments on all editions in the database. Its columns are the same as those in ALL_EDITION_COMMENTS.

**See Also:**

"ALL_EDITION_COMMENTS"

# DBA_EDITIONED_TYPES

DBA_EDITIONED_TYPES lists all types that are editioned by default for every user in the database.

### Related View

USER_EDITIONED_TYPES lists the types that are editioned by default for the current user. This view does not display the SCHEMA column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA | VARCHAR2(128) | NOT NULL | Schema in which the object types is editionable |
| OBJECT_TYPE | VARCHAR2(23) | | Object type that is editionable |

**See Also:**

"USER_EDITIONED_TYPES"

## DBA_EDITIONING_VIEW_COLS

DBA_EDITIONING_VIEW_COLS describes the relationship between the columns of all editioning views in the database and the table columns to which they map. Its columns are the same as those in ALL_EDITIONING_VIEW_COLS.

**See Also:**

"ALL_EDITIONING_VIEW_COLS"

## DBA_EDITIONING_VIEW_COLS_AE

DBA_EDITIONING_VIEW_COLS_AE describes the relationship between the columns of all editioning views (across all editions) in the database and the table columns to which they map. Its columns are the same as those in ALL_EDITIONING_VIEW_COLS_AE.

**See Also:**

"ALL_EDITIONING_VIEW_COLS_AE"

## DBA_EDITIONING_VIEWS

DBA_EDITIONING_VIEWS describes all editioning views in the database. Its columns are the same as those in ALL_EDITIONING_VIEWS.

**See Also:**

"ALL_EDITIONING_VIEWS"

## DBA_EDITIONING_VIEWS_AE

DBA_EDITIONING_VIEWS_AE describes all editioning views (across all editions) in the database. Its columns are the same as those in ALL_EDITIONING_VIEWS_AE.

**See Also:**

"ALL_EDITIONING_VIEWS_AE"

# DBA_EDITIONS

DBA_EDITIONS describes all editions in the database. Its columns are the same as those in ALL_EDITIONS.

**See Also:**

- "ALL_EDITIONS"
- *Oracle Database Development Guide* for more information about editions

# DBA_ENABLED_AGGREGATIONS

DBA_ENABLED_AGGREGATIONS displays information about enabled on-demand statistic aggregation.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AGGREGATION_TYPE | VARCHAR2(21) | | Type of the aggregation:<br>• CLIENT_ID<br>• SERVICE<br>• SERVICE_MODULE<br>• SERVICE_MODULE_ACTION |
| PRIMARY_ID | VARCHAR2(64) | | Primary qualifier (specific client identifier or service name) |
| QUALIFIER_ID1 | VARCHAR2(48) | | Secondary qualifier (specific module name) |
| QUALIFIER_ID2 | VARCHAR2(32) | | Additional qualifier (specific action name) |

# DBA_ENABLED_TRACES

DBA_ENABLED_TRACES displays information about enabled SQL traces.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRACE_TYPE | VARCHAR2(21) | | Type of the trace:<br>• CLIENT_ID<br>• SERVICE<br>• SERVICE_MODULE<br>• SERVICE_MODULE_ACTION<br>• DATABASE |
| PRIMARY_ID | VARCHAR2(64) | | Primary qualifier (specific client identifier or service name) |
| QUALIFIER_ID1 | VARCHAR2(64) | | Secondary qualifier (specific module name) |
| QUALIFIER_ID2 | VARCHAR2(64) | | Additional qualifier (specific action name) |
| WAITS | VARCHAR2(5) | | Indicates whether waits are traced (TRUE) or not (FALSE) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BINDS | VARCHAR2(5) | | Indicates whether binds are traced (TRUE) or not (FALSE) |
| INSTANCE_NAME | VARCHAR2(16) | | Instance name for tracing restricted to named instances |

**See Also:**

- The following support note on My Oracle Support at http://www.myoraclesupport.com for more information about using the DBMS_MONITOR PL/SQL package and the DBA_ENABLED_TRACES view:

  Note 293661.1 "Tracing Enhancements In 10g and 11g Using DBMS_MONITOR"

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_MONITOR package

# DBA_ENCRYPTED_COLUMNS

DBA_ENCRYPTED_COLUMNS maintains encryption algorithm information for all encrypted columns in the database. Its columns are the same as those in "ALL_ENCRYPTED_COLUMNS".

# DBA_EPG_DAD_AUTHORIZATION

DBA_EPG_DAD_AUTHORIZATION describes the DADs that are authorized to use different user's privileges.

### Related View

USER_EPG_DAD_AUTHORIZATION describes the DADs that are authorized to use the user's privileges. This view does not display the USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DAD_NAME | VARCHAR2(64) | NOT NULL | Name of DAD |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privileges the DAD is authorized to use |

**See Also:**

"USER_EPG_DAD_AUTHORIZATION"

# DBA_ERROR_TRANSLATIONS

DBA_ERROR_TRANSLATIONS describes all error translations in the database. Its columns are the same as those in ALL_ERROR_TRANSLATIONS.

**See Also:**

"ALL_ERROR_TRANSLATIONS"

## DBA_ERRORS

DBA_ERRORS describes the current errors on all stored objects in the database. Its columns are the same as those in ALL_ERRORS.

**See Also:**

"ALL_ERRORS"

## DBA_ERRORS_AE

DBA_ERRORS_AE describes the current errors on all stored objects (across all editions) in the database. Its columns are the same as those in ALL_ERRORS_AE.

**See Also:**

"ALL_ERRORS_AE"

## DBA_EVALUATION_CONTEXT_TABLES

DBA_EVALUATION_CONTEXT_TABLES describes the tables in all rule evaluation contexts in the database. Its columns are the same as those in ALL_EVALUATION_CONTEXT_TABLES.

**See Also:**

"ALL_EVALUATION_CONTEXT_TABLES"

## DBA_EVALUATION_CONTEXT_VARS

DBA_EVALUATION_CONTEXT_VARS describes the variables in all rule evaluation contexts in the database. Its columns are the same as those in ALL_EVALUATION_CONTEXT_VARS.

**See Also:**

"ALL_EVALUATION_CONTEXT_VARS"

## DBA_EVALUATION_CONTEXTS

DBA_EVALUATION_CONTEXTS describes all rule evaluation contexts in the database. Its columns are the same as those in ALL_EVALUATION_CONTEXTS.

**See Also:**

"ALL_EVALUATION_CONTEXTS"

# DBA_EXP_FILES

DBA_EXP_FILES describes export files.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXP_VERSION | NUMBER(3) | NOT NULL | Version number of the export session |
| EXP_TYPE | VARCHAR2(11) | | Type of export file: complete, cumulative, or incremental |
| FILE_NAME | VARCHAR2(100) | NOT NULL | Name of the export file |
| USER_NAME | VARCHAR2(128) | NOT NULL | Name of user who executed export |
| TIMESTAMP | DATE | NOT NULL | Timestamp of the export session |

# DBA_EXP_OBJECTS

DBA_EXP_OBJECTS describes objects that have been incrementally exported.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of exported object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of exported object |
| OBJECT_TYPE | VARCHAR2(13) | | Type of exported object |
| CUMULATIVE | DATE | | Timestamp of last cumulative export |
| INCREMENTAL | DATE | NOT NULL | Timestamp of last incremental export |
| EXPORT_VERSION | NUMBER(3) | NOT NULL | The ID of the export session |

# DBA_EXP_VERSION

DBA_EXP_VERSION displays the version number of the last export session.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXP_VERSION | NUMBER(3) | NOT NULL | Version number of the last export session |

# DBA_EXTENTS

DBA_EXTENTS describes the extents comprising the segments in all tablespaces in the database.

Note that if a data file (or entire tablespace) is offline in a locally managed tablespace, you will not see any extent information. If an object has extents in an online file of the tablespace, you will see extent information about the offline data file. However, if the object is entirely in the offline file, a query of this view will not return any records.

### Related View

USER_EXTENTS describes the extents comprising the segments owned by the current user's objects. This view does not display the OWNER, FILE_ID, BLOCK_ID, or RELATIVE_FNO columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the segment associated with the extent |
| SEGMENT_NAME | VARCHAR2(128) | | Name of the segment associated with the extent |
| PARTITION_NAME | VARCHAR2(128) | | Object Partition Name (Set to NULL for nonpartitioned objects) |
| SEGMENT_TYPE | VARCHAR2(18) | | Type of the segment: INDEX PARTITION, TABLE PARTITION |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the extent |
| EXTENT_ID | NUMBER | | Extent number in the segment |
| FILE_ID | NUMBER | | File identifier number of the file containing the extent |
| BLOCK_ID | NUMBER | | Starting block number of the extent |
| BYTES | NUMBER | | Size of the extent in bytes |
| BLOCKS | NUMBER | | Size of the extent in Oracle blocks |
| RELATIVE_FNO | NUMBER | | Relative file number of the first extent block |

**See Also:**

"USER_EXTENTS"

## DBA_EXTERNAL_LOCATIONS

DBA_EXTERNAL_LOCATIONS describes the locations (data sources) of all external tables in the database. Its columns are the same as those in ALL_EXTERNAL_LOCATIONS.

**See Also:**

"ALL_EXTERNAL_LOCATIONS"

## DBA_EXTERNAL_TABLES

DBA_EXTERNAL_TABLES describes all external tables in the database. Its columns are the same as those in ALL_EXTERNAL_TABLES.

**See Also:**

"ALL_EXTERNAL_TABLES"

# DBA_FEATURE_USAGE_STATISTICS

DBA_FEATURE_USAGE_STATISTICS displays information about database feature usage statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database identifier of the database being tracked |
| NAME | VARCHAR2(64) | NOT NULL | Name of the feature |
| VERSION | VARCHAR2(17) | NOT NULL | Database version in which the feature was tracked |
| DETECTED_USAGES | NUMBER | NOT NULL | Number of times the system has detected usage for the feature |
| TOTAL_SAMPLES | NUMBER | NOT NULL | Number of times the system has woken up and checked for feature usage |
| CURRENTLY_USED | VARCHAR2(5) | | Indicates whether usage was detected the last time the system checked (TRUE) or not (FALSE) |
| FIRST_USAGE_DATE | DATE | | First sample time the system detected usage of the feature |
| LAST_USAGE_DATE | DATE | | Last sample time the system detected usage of the feature |
| AUX_COUNT | NUMBER | | This column stores feature-specific usage data in number format. |
| FEATURE_INFO | CLOB | | This column stores feature-specific usage data in character format. |
| LAST_SAMPLE_DATE | DATE | | Amount of time (in seconds) between the last two usage sample times |
| LAST_SAMPLE_PERIOD | NUMBER | | Amount of time (in hours) between the last two usage sample times |
| SAMPLE_INTERVAL | NUMBER | | Sample interval |
| DESCRIPTION | VARCHAR2(128) | | Description of the feature and usage detection logic |

**Note:**

Use the following SQL query to list the database features and their descriptions in alphabetical order:

```
SELECT name, description FROM dba_feature_usage_statistics
ORDER BY name;
```

# DBA_FGA_AUDIT_TRAIL

DBA_FGA_AUDIT_TRAIL displays all audit records for fine-grained auditing.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SESSION_ID | NUMBER | NOT NULL | Session id of the query |
| TIMESTAMP | DATE | | Date and time of the query in the local database session time zone |
| DB_USER | VARCHAR2(128) | | Database username who executed the query |
| OS_USER | VARCHAR2(255) | | Operating system username who executed the query |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| CLIENT_ID | VARCHAR2(128) | | Client identifier in each Oracle session |
| ECONTEXT_ID | VARCHAR2(64) | | Application execution context identifier |
| EXT_NAME | VARCHAR2(4000) | | External name |
| OBJECT_SCHEMA | VARCHAR2(128) | | Owner of the table or view |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or view |
| POLICY_NAME | VARCHAR2(128) | | Name of the Fine-Grained Auditing Policy |
| SCN | NUMBER | | System change number (SCN) of the query |
| SQL_TEXT | NVARCHAR2(2000) | | SQL text of the query |
| SQL_BIND | NVARCHAR2(2000) | | Bind variable data of the query |
| COMMENT$TEXT | VARCHAR2(4000) | | Comments |
| STATEMENT_TYPE | VARCHAR2(7) | | Statement type of the query:<br>• SELECT<br>• INSERT<br>• UPDATE<br>• DELETE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of the query in UTC (Coordinated Universal Time) time zone |
| PROXY_SESSIONID | NUMBER | | Proxy session serial number, if an enterprise user has logged in through the proxy mechanism |
| GLOBAL_UID | VARCHAR2(32) | | Global user identifier for the user, if the user has logged in as an enterprise user |
| INSTANCE_NUMBER | NUMBER | | Instance number as specified by the INSTANCE_NUMBER initialization parameter |
| OS_PROCESS | VARCHAR2(16) | | Operating System process identifier of the Oracle process |
| TRANSACTIONID | RAW(8) | | Transaction identifier of the transaction in which the object is accessed or modified |
| STATEMENTID | NUMBER | | Numeric ID for each statement run (a statement may cause many actions) |
| ENTRYID | NUMBER | | Numeric ID for each audit trail entry in the session |
| OBJ_EDITION_NAME | VARCHAR2(128) | | Name of the edition containing the audited object |
| DBID | NUMBER | | Database identifier of the audited database |

**Note:**

The SQL_BIND and SQL_TEXT columns are populated only if the policy has been created with the AUDIT_TRAIL parameter set to db, extended.

# DBA_FILE_GROUP_EXPORT_INFO

DBA_FILE_GROUP_EXPORT_INFO shows export-related information for each version in the database that has a valid Data Pump dump file. Its columns are the same as those in "ALL_FILE_GROUP_EXPORT_INFO".

# DBA_FILE_GROUP_FILES

DBA_FILE_GROUP_FILES shows the file set for each versioned file group in the database. Its columns are the same as those in "ALL_FILE_GROUP_FILES".

# DBA_FILE_GROUP_TABLES

DBA_FILE_GROUP_TABLES shows information about all the tables in the database that can be imported using the file set. Its columns are the same as those in "ALL_FILE_GROUP_TABLES".

# DBA_FILE_GROUP_TABLESPACES

DBA_FILE_GROUP_TABLESPACES shows information about the transportable tablespaces present (partially or completely) in all file sets in the database (when the file set contains dump files). Its columns are the same as those in "ALL_FILE_GROUP_TABLESPACES".

# DBA_FILE_GROUP_VERSIONS

DBA_FILE_GROUP_VERSIONS shows top-level version information for all file groups in the database. Its columns are the same as those in "ALL_FILE_GROUP_VERSIONS".

# DBA_FILE_GROUPS

DBA_FILE_GROUPS shows top-level metadata about all file groups in the database. Its columns are the same as those for "ALL_FILE_GROUPS".

# DBA_FLASHBACK_ARCHIVE

DBA_FLASHBACK_ARCHIVE describes all flashback archives available in the database.

### Related View

USER_FLASHBACK_ARCHIVE describes the flashback archives available to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER_NAME | VARCHAR2(30) | | Name of the creator of the flashback archive |
| FLASHBACK_ARCHIVE_NAME | VARCHAR2(255) | NOT NULL | Name of the flashback archive |
| FLASHBACK_ARCHIVE# | NUMBER | NOT NULL | Number of the flashback archive |
| RETENTION_IN_DAYS | NUMBER | NOT NULL | Maximum duration (in days) for which data is retained in the flashback archive |
| CREATE_TIME | TIMESTAMP(9) | | Time at which the flashback archive was created |
| LAST_PURGE_TIME | TIMESTAMP(9) | | Time at which the data in the flashback archive was last purged by the system |
| STATUS | VARCHAR2(7) | | Indicates whether the flashback archive is a default flashback archive for the system (DEFAULT) or not (NULL) |

**See Also:**

"USER_FLASHBACK_ARCHIVE"

# DBA_FLASHBACK_ARCHIVE_TABLES

DBA_FLASHBACK_ARCHIVE_TABLES displays information about all tables in the database that are enabled for Flashback Archive.

### Related View

USER_FLASHBACK_ARCHIVE_TABLES displays information about the tables owned by the current user that are enabled for Flashback Archive.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table enabled for Flashback Archive |
| OWNER_NAME | VARCHAR2(128) | NOT NULL | Owner name of the table enabled for Flashback Archive |
| FLASHBACK_ARCHIVE_NAME | VARCHAR2(255) | NOT NULL | Name of the flashback archive |
| ARCHIVE_TABLE_NAME | VARCHAR2(53) | | Name of the archive table containing the historical data for the user table |
| STATUS | VARCHAR2(13) | | Status of whether flashback archive is enabled or being disabled on the table |

**See Also:**

"USER_FLASHBACK_ARCHIVE_TABLES"

# DBA_FLASHBACK_ARCHIVE_TS

DBA_FLASHBACK_ARCHIVE_TS describes all tablespaces in the flashback archives available in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FLASHBACK_ARCHIVE_NAME | VARCHAR2(255) | NOT NULL | Name of the flashback archive |
| FLASHBACK_ARCHIVE# | NUMBER | NOT NULL | Number of the flashback archive |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of a tablespace in the flashback archive |
| QUOTA_IN_MB | VARCHAR2(40) | | Maximum space (in MB) that can be used for Flashback Archive from the tablespace; NULL indicates no Quota restriction |

# DBA_FLASHBACK_TXN_REPORT

DBA_FLASHBACK_TXN_REPORT displays information about all compensating transactions that have been committed in the database. Each row in this view is associated with one compensating transaction.

### Related View

USER_FLASHBACK_TXN_REPORT displays information about the compensating transactions owned by the current user that have been committed in the database. This view does not display the USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPENSATING_XID | RAW(8) | NOT NULL | Transaction responsible for backout |
| COMPENSATING_TXN_NAME | VARCHAR2(256) | | Name of the compensating transaction |
| COMMIT_TIME | DATE | | Timestamp when the compensating transaction committed |
| XID_REPORT | CLOB | | An XML report describing the details of the transactions backed out by the compensating transaction |
| USERNAME | VARCHAR2(128) | NOT NULL | User who is executing the compensating transaction |

**See Also:**

"USER_FLASHBACK_TXN_REPORT"

## DBA_FLASHBACK_TXN_STATE

DBA_FLASHBACK_TXN_STATE displays information about the compensating status of all transactions in the database. For each compensating transaction, there could be multiple rows, where each row provides the dependency relation between the transactions that have been compensated by the compensating transaction.

### Related View

USER_FLASHBACK_TXN_STATE displays information about the compensating status of the transactions owned by the current user. This view does not display the USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPENSATING_XID | RAW(8) | | Transaction ID of the compensating transaction |
| XID | RAW(8) | | A transaction that has been compensated by the compensating transaction |
| DEPENDENT_XID | RAW(8) | | A dependent transaction of XID<br>**Note:** In the case of BACKOUT_MODE = CASCADE, there must be another row with XID = DEPENDENT_XID of this column. |
| BACKOUT_MODE | VARCHAR2(16) | | Mode in which XID was backed out:<br>• NOCASCADE<br>• NOCASCADE_FORCE<br>• NONCONFLICT_ONLY<br>• CASCADE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | NOT NULL | User who is performing the compensating transaction |

**See Also:**

"USER_FLASHBACK_TXN_STATE"

# DBA_FREE_SPACE

DBA_FREE_SPACE describes the free extents in all tablespaces in the database.

Note that if a data file (or entire tablespace) is offline in a locally managed tablespace, you will not see any extent information. If an object has extents in an online file of the tablespace, you will see extent information about the offline data file. However, if the object is entirely in the offline file, a query of this view will not return any records.

### Related View

USER_FREE_SPACE describes the free extents in the tablespaces accessible to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the extent |
| FILE_ID | NUMBER | | File identifier number of the file containing the extent |
| BLOCK_ID | NUMBER | | Starting block number of the extent |
| BYTES | NUMBER | | Size of the extent (in bytes) |
| BLOCKS | NUMBER | | Size of the extent (in Oracle blocks) |
| RELATIVE_FNO | NUMBER | | Relative file number of the file containing the extent |

**See Also:**

"USER_FREE_SPACE"

# DBA_FREE_SPACE_COALESCED

DBA_FREE_SPACE_COALESCED describes statistics on coalesced space in all tablespaces in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace |
| TOTAL_EXTENTS | NUMBER | | Total number of free extents in the tablespace |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXTENTS_COALESCED | NUMBER | | Total number of coalesced free extents in the tablespace |
| PERCENT_EXTENTS_COALESCED | NUMBER | | Percentage of coalesced free extents in the tablespace |
| TOTAL_BYTES | NUMBER | | Total number of free bytes in the tablespace |
| BYTES_COALESCED | NUMBER | | Total number of coalesced free bytes in the tablespace |
| TOTAL_BLOCKS | NUMBER | | Total number of free Oracle blocks in the tablespace |
| BLOCKS_COALESCED | NUMBER | | Total number of coalesced free Oracle blocks in the tablespace |
| PERCENT_BLOCKS_COALESCED | NUMBER | | Percentage of coalesced free Oracle blocks in the tablespace |

# DBA_GG_INBOUND_PROGRESS

DBA_GG_INBOUND_PROGRESS displays information about the progress made by all GoldenGate inbound servers in the database. Its columns are the same as those in ALL_GG_INBOUND_PROGRESS.

**See Also:**

"ALL_GG_INBOUND_PROGRESS"

# DBA_GLOBAL_CONTEXT

DBA_GLOBAL_CONTEXT displays the definition (name, schema, and package) of all global contexts created in the database. This view is a subset of DBA_CONTEXT, which describes all contexts, including global contexts.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAMESPACE | VARCHAR2(128) | NOT NULL | Name of the context namespace |
| SCHEMA | VARCHAR2(128) | NOT NULL | Schema of the package that administers the globally accessible context |
| PACKAGE | VARCHAR2(128) | NOT NULL | Package that administers the globally accessible context |

**See Also:**

- "DBA_CONTEXT"
- *Oracle Database Security Guide* for more information about using global application contexts
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_SESSION.SET_CONTEXT` procedure

## DBA_GOLDENGATE_INBOUND

`DBA_GOLDENGATE_INBOUND` displays information about all GoldenGate inbound servers in the database. Its columns are the same as those in `ALL_GOLDENGATE_INBOUND`.

**See Also:**

"ALL_GOLDENGATE_INBOUND"

## DBA_GOLDENGATE_PRIVILEGES

`DBA_GOLDENGATE_PRIVILEGES` displays details about Oracle GoldenGate privileges. Its columns are the same as those in `ALL_GOLDENGATE_PRIVILEGES`.

**See Also:**

"ALL_GOLDENGATE_PRIVILEGES"

## DBA_GOLDENGATE_RULES

`DBA_GOLDENGATE_RULES` displays information about all GoldenGate server rules in the database. Its columns are the same as those in `ALL_GOLDENGATE_RULES`.

**See Also:**

"ALL_GOLDENGATE_RULES"

## DBA_GOLDENGATE_SUPPORT_MODE

`DBA_GOLDENGATE_SUPPORT_MODE` displays information about the level of Oracle GoldenGate capture process support for the tables in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `OWNER` | `VARCHAR2(128)` | | Table owner |
| `OBJECT_NAME` | `VARCHAR2(128)` | | Table name |
| `SUPPORT_MODE` | `VARCHAR2(6)` | | Capture process support level for the table: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • FULL - A capture process can capture changes made to all of the columns in the table<br>• ID KEY - A capture process can capture changes made to the key columns and any other columns in the table supported by the capture process, except for LOB, LONG, LONG RAW, and XMLType columns.<br>• NONE - A capture process cannot capture changes made to any columns in the table. |

# DBA_HEAT_MAP_SEG_HISTOGRAM

DBA_HEAT_MAP_SEG_HISTOGRAM displays segment access information for all segments. Its columns are the same as those in ALL_HEAT_MAP_SEG_HISTOGRAM.

**See Also:**

"ALL_HEAT_MAP_SEG_HISTOGRAM"

# DBA_HEAT_MAP_SEGMENT

DBA_HEAT_MAP_SEGMENT displays the latest segment access time for all segments. Its columns are the same as those in ALL_HEAT_MAP_SEGMENT.

**See Also:**

"ALL_HEAT_MAP_SEGMENT"

# DBA_HEATMAP_TOP_OBJECTS

DBA_HEATMAP_TOP_OBJECTS displays heat map information for the top 10000 objects by default. If the database contains fewer than 10000 objects, then fewer than 10000 objects are returned by the view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Object name |
| OBJECT_TYPE | VARCHAR2(18) | | Object type |
| TABLESPACE_NAME | VARCHAR2(30) | | Tablespace name |
| SEGMENT_COUNT | NUMBER | | Segments in the tablespace |
| OBJECT_SIZE | NUMBER | | Size of the object in MB |
| MIN_WRITETIME | DATE | | Minimum write time of the object |
| MAX_WRITETIME | DATE | | Maximum write time of the object |
| AVG_WRITETIME | DATE | | Average write time of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MIN_READTIME | DATE | | Minimum read time of the object |
| MAX_READTIME | DATE | | Maximum read time of the object |
| AVG_READTIME | DATE | | Average read time of the object |
| MIN_FTSTIME | DATE | | Minimum full table scan time of the object |
| MAX_FTSTIME | DATE | | Maximum full table scan time of the object |
| AVG_FTSTIME | DATE | | Average full table scan time of the object |
| MIN_LOOKUPTIME | DATE | | Minimum lookup time of the object |
| MAX_LOOKUPTIME | DATE | | Maximum lookup time of the object |
| AVG_LOOKUPTIME | DATE | | Average lookup time of the object |

# DBA_HEATMAP_TOP_TABLESPACES

DBA_HEATMAP_TOP_TABLESPACES displays heat map information for the top 100 tablespaces.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(128) | | Tablespace name |
| SEGMENT_COUNT | NUMBER | | Segments in the tablespace |
| ALLOCATED_BYTES | NUMBER | | Total bytes allocated to the objects in the tablespace |
| MIN_WRITETIME | DATE | | Minimum write time of objects tracked |
| MAX_WRITETIME | DATE | | Maximum write time of objects tracked |
| AVG_WRITETIME | DATE | | Average write time of objects tracked |
| MIN_READTIME | DATE | | Minimum read time of objects tracked |
| MAX_READTIME | DATE | | Maximum read time of objects tracked |
| AVG_READTIME | DATE | | Average read time of objects tracked |
| MIN_FTSTIME | DATE | | Minimum full table scan time of objects tracked |
| MAX_FTSTIME | DATE | | Maximum full table scan time of objects tracked |
| AVG_FTSTIME | DATE | | Average full table scan time of objects tracked |
| MIN_LOOKUPTIME | DATE | | Minimum lookup time of objects tracked |
| MAX_LOOKUPTIME | DATE | | Maximum lookup time of objects tracked |
| AVG_LOOKUPTIME | DATE | | Average lookup time of objects tracked |

# DBA_HIGH_WATER_MARK_STATISTICS

DBA_HIGH_WATER_MARK_STATISTICS displays information about database high-watermark statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Identifier of the database for which the high-watermark statistics are tracked |
| NAME | VARCHAR2(64) | NOT NULL | Name of the high-watermark statistic (see Table 4-1) |
| VERSION | VARCHAR2(17) | NOT NULL | Database version in which the high watermarks were tracked |
| HIGHWATER | NUMBER | | Highest value of the statistic seen at sampling time |
| LAST_VALUE | NUMBER | | Value of the statistic at the last sample time |
| DESCRIPTION | VARCHAR2(128) | | Description of the high-watermark statistics (see Table 4-1) |

***Table 4-1 DBA_HIGH_WATER_MARK_STATISTICS Statistics***

| Name | Description |
|---|---|
| ACTIVE_SESSIONS | Maximum Number of Active Sessions seen in the system |
| CPU_COUNT | Maximum Number of CPUs |
| DATAFILES | Maximum Number of Datafiles |
| DB_SIZE | Maximum Size of the Database (Bytes) |
| EXADATA_DISKS | Number of physical disks |
| INSTANCES | Oracle Database instances |
| PART_INDEXES | Maximum Number of Partitions belonging to an User Index |
| PART_TABLES | Maximum Number of Partitions belonging to an User Table |
| QUERY_LENGTH | Maximum Query Length |
| SEGMENT_SIZE | Size of Largest Segment (Bytes) |
| SESSIONS | Maximum Number of Concurrent Sessions seen in the database |
| SQL_NCHAR_COLUMNS | Maximum Number of SQL NCHAR Columns |
| TABLESPACES | Maximum Number of Tablespaces |
| USER_INDEXES | Number of User Indexes |
| USER_MV | Maximum Number of Materialized Views (User) |
| USER_TABLES | Number of User Tables |

# DBA_HIST_ACTIVE_SESS_HISTORY

DBA_HIST_ACTIVE_SESS_HISTORY displays the history of the contents of the in-memory active session history of recent system activity. This view contains snapshots of V$ACTIVE_SESSION_HISTORY. See "V$ACTIVE_SESSION_HISTORY" for further interpretation details for many of these columns (except SNAP_ID, DBID, and INSTANCE_NUMBER).

**Note:**

If you want to perform a join with the snapshots view, use the DBA_HIST_ASH_SNAPSHOT view instead of the DBA_HIST_SNAPSHOT view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SAMPLE_ID | NUMBER | NOT NULL | ID of the sample |
| SAMPLE_TIME | TIMESTAMP(3) | NOT NULL | Time of the sample |
| SESSION_ID | NUMBER | NOT NULL | Session identifier |
| SESSION_SERIAL# | NUMBER | | Session serial number (used to uniquely identify a session's objects) |
| SESSION_TYPE | VARCHAR2(10) | | Session type:<br>• FOREGROUND<br>• BACKGROUND |
| FLAGS | NUMBER | | Reserved for future use |
| USER_ID | NUMBER | | Oracle user identifier |
| SQL_ID | VARCHAR2(13) | | SQL identifier of the SQL statement that is currently being executed |
| IS_SQLID_CURRENT | VARCHAR2(1) | | Indicates whether the SQL identifier in the SQL_ID column is being executed (Y) or not (N) |
| SQL_CHILD_NUMBER | NUMBER | | Child number of the SQL statement that is currently being executed |
| SQL_OPCODE | NUMBER | | Indicates what phase of operation the SQL statement is in |
| SQL_OPNAME | VARCHAR2(64) | | SQL command name |
| FORCE_MATCHING_SIGNATURE | NUMBER | | Signature used when the CURSOR_SHARING parameter is set to FORCE |
| TOP_LEVEL_SQL_ID | VARCHAR2(13) | | SQL identifier of the top level SQL statement |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TOP_LEVEL_SQL_OPCODE | NUMBER | | Indicates what phase of operation the top level SQL statement was in |
| SQL_PLAN_HASH_VALUE | NUMBER | | Numerical representation of the SQL plan for the cursor |
| SQL_FULL_PLAN_HASH_VALUE | NUMBER | | Numerical representation of the complete SQL plan for the cursor being executed by this session |
| SQL_ADAPTIVE_PLAN_RESOLVED | NUMBER | | Indicates whether the SQL plan of the sampled database session is a resolved adaptive plan or not |
| SQL_PLAN_LINE_ID | NUMBER | | SQL plan line ID |
| SQL_PLAN_OPERATION | VARCHAR2(64) | | Plan operation name |
| SQL_PLAN_OPTIONS | VARCHAR2(64) | | Plan operation options |
| SQL_EXEC_ID | NUMBER | | SQL execution identifier |
| SQL_EXEC_START | DATE | | Time when the execution of the SQL started |
| PLSQL_ENTRY_OBJECT_ID | NUMBER | | Object ID of the top-most PL/SQL subprogram on the stack (or NULL if there is no PL/SQL subprogram on the stack) |
| PLSQL_ENTRY_SUBPROGRAM_ID | NUMBER | | Subprogram ID of the top-most PL/SQL subprogram on the stack (or NULL if there is no PL/SQL subprogram on the stack) |
| PLSQL_OBJECT_ID | NUMBER | | Object ID of the currently executing PL/SQL subprogram (or NULL if executing SQL) |
| PLSQL_SUBPROGRAM_ID | NUMBER | | Subprogram ID of the currently executing PL/SQL object (or NULL if executing SQL) |
| QC_INSTANCE_ID | NUMBER | | Query coordinator instance ID |
| QC_SESSION_ID | NUMBER | | Query coordinator session ID |
| QC_SESSION_SERIAL# | NUMBER | | Query coordinator session serial number |
| PX_FLAGS | NUMBER | | Reserved for internal use |
| EVENT | VARCHAR2(64) | | If `SESSION_STATE = WAITING`, then the event for which the session was waiting at the time of sampling.<br>If `SESSION_STATE = ON CPU`, then this column will be NULL. |
| EVENT_ID | NUMBER | | Identifier of the resource or event for which the session is waiting or for which the session last waited |
| SEQ# | NUMBER | | Sequence number that uniquely identifies the wait (incremented for each wait) |
| P1TEXT | VARCHAR2(64) | | Text of first additional parameter |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| P1 | NUMBER | | First additional parameter |
| P2TEXT | VARCHAR2(64) | | Text of second additional parameter |
| P2 | NUMBER | | Second additional parameter |
| P3TEXT | VARCHAR2(64) | | Text of third additional parameter |
| P3 | NUMBER | | Third additional parameter |
| WAIT_CLASS | VARCHAR2(64) | | Wait class name of the event for which the session was waiting at the time of sampling. Interpretation is similar to that of the EVENT column. Maps to V$SESSION.WAIT_CLASS. |
| WAIT_CLASS_ID | NUMBER | | Wait class identifier of the event for which the session was waiting at the time of sampling. Interpretation is similar to that of the EVENT column. Maps to V$SESSION.WAIT_CLASS_ID. |
| WAIT_TIME | NUMBER | | Total wait time (in microseconds) for the event for which the session last waited (0 if currently waiting) |
| SESSION_STATE | VARCHAR2(7) | | Session state:<br>• WAITING<br>• ON CPU |
| TIME_WAITED | NUMBER | | Time that the current session actually spent waiting for the event (in microseconds). This column is set for waits that were in progress at the time the sample was taken. |
| BLOCKING_SESSION_STATUS | VARCHAR2(11) | | Status of the blocking session:<br>• VALID<br>• NO HOLDER<br>• GLOBAL<br>• NOT IN WAIT<br>• UNKNOWN |
| BLOCKING_SESSION | NUMBER | | Session identifier of the blocking session. Populated only when the session was waiting for enqueues or a "buffer busy" wait. Maps to V$SESSION.BLOCKING_SESSION. |
| BLOCKING_SESSION_SERIAL# | NUMBER | | Serial number of the blocking session |
| BLOCKING_INST_ID | NUMBER | | Instance number of the blocker shown in BLOCKING_SESSION |
| BLOCKING_HANGCHAIN_INFO | VARCHAR2(1) | | Indicates whether the information about BLOCKING_SESSION comes from the hang chain (Y) or not (N) |
| CURRENT_OBJ# | NUMBER | | Object ID of the object that the session is currently referencing. This information is only available if the session was waiting for Application, Cluster, Concurrency, and User I/O |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | wait events. Maps to V $SESSION.ROW_WAIT_OBJ#. |
| CURRENT_FILE# | NUMBER | | File number of the file containing the block that the session is currently referencing. This information is only available if the session was waiting for Cluster, Concurrency, and User I/O wait events. Maps to V $SESSION.ROW_WAIT_FILE#. |
| CURRENT_BLOCK# | NUMBER | | ID of the block that the session is currently referencing |
| CURRENT_ROW# | NUMBER | | Row identifier that the session is referencing |
| TOP_LEVEL_CALL# | NUMBER | | Oracle top level call number |
| TOP_LEVEL_CALL_NAME | VARCHAR2(64) | | Oracle top level call name |
| CONSUMER_GROUP_ID | NUMBER | | Consumer group ID |
| XID | RAW(8) | | Transaction ID that the session was working on at the time of sampling. V$SESSION does not contain this information. |
| REMOTE_INSTANCE# | NUMBER | | Remote instance identifier that will serve the block that this session is waiting for. This information is only available if the session was waiting for cluster events. |
| TIME_MODEL | NUMBER | | Time model information |
| IN_CONNECTION_MGMT | VARCHAR2(1) | | Indicates whether the session was doing connection management at the time of sampling (Y) or not (N) |
| IN_PARSE | VARCHAR2(1) | | Indicates whether the session was parsing at the time of sampling (Y) or not (N) |
| IN_HARD_PARSE | VARCHAR2(1) | | Indicates whether the session was hard parsing at the time of sampling (Y) or not (N) |
| IN_SQL_EXECUTION | VARCHAR2(1) | | Indicates whether the session was executing SQL statements at the time of sampling (Y) or not (N) |
| IN_PLSQL_EXECUTION | VARCHAR2(1) | | Indicates whether the session was executing PL/SQL at the time of sampling (Y) or not (N) |
| IN_PLSQL_RPC | VARCHAR2(1) | | Indicates whether the session was executing inbound PL/SQL RPC calls at the time of sampling (Y) or not (N) |
| IN_PLSQL_COMPILATION | VARCHAR2(1) | | Indicates whether the session was compiling PL/SQL at the time of sampling (Y) or not (N) |
| IN_JAVA_EXECUTION | VARCHAR2(1) | | Indicates whether the session was executing Java at the time of sampling (Y) or not (N) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| IN_BIND | VARCHAR2(1) | | Indicates whether the session was doing bind operations at the time of sampling (Y) or not (N) |
| IN_CURSOR_CLOSE | VARCHAR2(1) | | Indicates whether the session was closing a cursor at the time of sampling (Y) or not (N) |
| IN_INMEMORY_QUERY | VARCHAR2(1) | | Indicates whether the session was querying the In-Memory Column Store (IM column store) at the time of sampling (Y) or not (N) |
| IN_INMEMORY_POPULATE[1] | VARCHAR2(1) | | Indicates whether the session was populating the IM column store at the time of sampling (Y) or not (N) |
| IN_INMEMORY_PREPOPULATE[1] | VARCHAR2(1) | | Indicates whether the session was prepopulating the IM column store at the time of sampling (Y) or not (N) |
| IN_INMEMORY_REPOPULATE[1] | VARCHAR2(1) | | Indicates whether the session was repopulating the IM column store at the time of sampling (Y) or not (N) |
| IN_INMEMORY_TREPOPULATE[1] | VARCHAR2(1) | | Indicates whether the session was trickle repopulating the IM column store at the time of sampling (Y) or not (N) |
| IN_SEQUENCE_LOAD | VARCHAR2(1) | | Indicates whether the session is loading in sequence (in sequence load code) (Y) or not (N) |
| CAPTURE_OVERHEAD | VARCHAR2(1) | | Indicates whether the session is executing capture code (Y) or not (N) |
| REPLAY_OVERHEAD | VARCHAR2(1) | | Indicates whether the session is executing replay code (Y) or not (N) |
| IS_CAPTURED | VARCHAR2(1) | | Indicates whether the session is being captured (Y) or not (N) |
| IS_REPLAYED | VARCHAR2(1) | | Indicates whether the session is being replayed (Y) or not (N) |
| SERVICE_HASH | NUMBER | | Hash that identifies the Service |
| PROGRAM | VARCHAR2(64) | | Name of the operating system program |
| MODULE | VARCHAR2(64) | | Name of the currently executing module as set by the DBMS_APPLICATION_INFO.SET_MODULE procedure |
| ACTION | VARCHAR2(64) | | Name of the currently executing action as set by the DBMS_APPLICATION_INFO.SET_ACTION procedure |
| CLIENT_ID | VARCHAR2(64) | | Client identifier of the session |
| MACHINE | VARCHAR2(64) | | Client's operating system machine name |
| PORT | NUMBER | | Client port number |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ECID | VARCHAR2(64) | | Execution context identifier (sent by Application Server) |
| DBREPLAY_FILE_ID | NUMBER | | If the session is being captured or replayed, then DBREPLAY_FILE_ID is the file ID for the workload capture or workload replay; otherwise it is NULL. |
| DBREPLAY_CALL_COUNTER | NUMBER | | If the session is being captured or replayed, then DBREPLAY_CALL_COUNTER is the call counter of the user call that is being captured or replayed; otherwise it is NULL. |
| TM_DELTA_TIME | NUMBER | | Time interval (in microseconds) over which TM_DELTA_CPU_TIME and TM_DELTA_DB_TIME are accumulated |
| TM_DELTA_CPU_TIME | NUMBER | | Amount of time this session spent on CPU over the last TM_DELTA_TIME microseconds |
| TM_DELTA_DB_TIME | NUMBER | | Amount of time spent by this session in database calls over the last TM_DELTA_TIME microseconds |
| DELTA_TIME | NUMBER | | Time interval (in microseconds) since the last time this session was sampled or created, over which the next five statistics are accumulated |
| DELTA_READ_IO_REQUESTS | NUMBER | | Number of read I/O requests made by this session over the last DELTA_TIME microseconds |
| DELTA_WRITE_IO_REQUESTS | NUMBER | | Number of write I/O requests made by this session over the last DELTA_TIME microseconds |
| DELTA_READ_IO_BYTES | NUMBER | | Number of I/O bytes read by this session over the last DELTA_TIME microseconds |
| DELTA_WRITE_IO_BYTES | NUMBER | | Number of I/O bytes written by this session over the last DELTA_TIME microseconds |
| DELTA_INTERCONNECT_IO_BYTES | NUMBER | | Number of I/O bytes sent over the I/O interconnect over the last DELTA_TIME microseconds |
| PGA_ALLOCATED | NUMBER | | Amount of PGA memory (in bytes) consumed by this session at the time this sample was taken |
| TEMP_SPACE_ALLOCATED | NUMBER | | Amount of TEMP memory (in bytes) consumed by this session at the time this sample was taken |
| DBOP_NAME | VARCHAR2(64) | | Database operation name. If the type is SQL, the DBOP_NAME will be NULL. |
| DBOP_EXEC_ID | NUMBER | | Database operation execution identifier for the current execution. If the type is SQL, the DBOP_EXEC_ID will be NULL. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

1 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_HIST_ASH_SNAPSHOT"
- "DBA_HIST_SNAPSHOT"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO package

# DBA_HIST_APPLY_SUMMARY

DBA_HIST_APPLY_SUMMARY displays historical statistics information about each apply process for Oracle Streams, Oracle GoldenGate, and Oracle XStream. This view is intended for use with Automatic Workload Repository (AWR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| APPLY_NAME | VARCHAR2(128) | NOT NULL | Name of the apply process |
| STARTUP_TIME | DATE | NOT NULL | The time that the apply process was last started |
| READER_TOTAL_MESSAGES_DEQUEUED | NUMBER | | Total number of messages dequeued since the apply process was last started |
| READER_LAG | NUMBER | | For captured messages, the delay (in seconds) between the creation of the last message and it being received by the apply process. For user enqueued messages, the delay between the message being enqueued in the local database and being received by the apply process. |
| COORD_TOTAL_RECEIVED | NUMBER | | Total number of transactions received by the coordinator process since the apply process was last started |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COORD_TOTAL_APPLIED | NUMBER | | Total number of transactions applied by the apply process since the apply process was last started |
| COORD_TOTAL_ROLLBACKS | NUMBER | | Number of transactions which were rolled back due to unexpected contention |
| COORD_TOTAL_WAIT_DEPS | NUMBER | | Number of times since the apply process was last started that an apply server waited to apply a logical change record (LCR) in a transaction until another apply server applied a transaction because of a dependency between the transactions |
| COORD_TOTAL_WAIT_CMTS | NUMBER | | Number of times since the apply process was last started that an apply server waited to commit a transaction until another apply server committed a transaction to serialize commits |
| COORD_LWM_LAG | NUMBER | | For captured messages, the delay (in seconds) between the creation of the message corresponding to the low watermark and it being applied by the apply process. For user enqueued messages, the delay between the message being enqueued in the local database and being applied by the apply process. |
| SERVER_TOTAL_MESSAGES_APPLIED | NUMBER | | Total number of messages applied by all the apply servers since the apply process was last started |
| SERVER_ELAPSED_DEQUEUE_TIME | NUMBER | | Time elapsed (in hundredths of a second) dequeuing messages by all the apply servers since the apply process was last started |
| SERVER_ELAPSED_APPLY_TIME | NUMBER | | Time elapsed (in hundredths of a second) applying messages by all the apply servers since the apply process was last started |
| CON_DBID | NUMBER | | The database ID of the PDB |
| REPLICAT_NAME | VARCHAR2(128) | | The name of the replicat group created from GGSCI using GoldenGate |
| UNASSIGNED_COMPLETE_TXN | NUMBER | | Total number of complete transactions that the coordinator has not assigned to any apply servers |
| TOTAL_LCRS_RETRIED | NUMBER | | Total number of LCRs retried by this server |
| TOTAL_TRANSACTIONS_RETRIED | NUMBER | | Total transactions retried by this server |
| TOTAL_ERRORS | NUMBER | | Number of transactions applied by the apply process that resulted in an apply error since the apply process was last started |
| SESSION_MODULE | VARCHAR2(64) | NOT NULL | Session module. Valid values:<br>• Streams<br>• XStream |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • GoldenGate |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_ASH_SNAPSHOT

DBA_HIST_ASH_SNAPSHOT provides the list of snapshots that contains Active Session History (ASH) data. This view differs from DBA_HIST_SNAPSHOT in that it provides snapshots which had errors flushing some Automatic Workload Repository (AWR) tables, but for which ASH data may be successfully flushed (DBA_HIST_SNAPSHOT filters out snapshots which had errors flushing AWR tables).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| STARTUP_TIME | TIMESTAMP(3) | NOT NULL | Startup time of the instance |
| BEGIN_INTERVAL_TIME | TIMESTAMP(3) | NOT NULL | Time at the beginning of the snapshot interval |
| END_INTERVAL_TIME | TIMESTAMP(3) | NOT NULL | Time at the end of the snapshot interval; the actual time the snapshot was taken |
| FLUSH_ELAPSED | INTERVAL DAY(5) TO SECOND(1) | | Amount of time to perform the snapshot |
| SNAP_LEVEL | NUMBER | | Snapshot level |
| STATUS | NUMBER | | Indicates if the snapshot was successfully flushed without any errors. Possible values:<br>• 0 - No errors<br>• 1 - Errors on some AWR tables |
| ERROR_COUNT | NUMBER | | Number of errors occurring in the tables for the particular snapshot |
| BL_MOVED | NUMBER | | Reserved for internal use |
| SNAP_FLAG | NUMBER | | Condition under which the snapshot was inserted. Possible values are:<br>• 1 - Manual snapshot created using a PL/SQL package<br>• 2 - Imported snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 4 - Snapshot taken while Diagnostic Pack or Tuning Pack was not enabled |
| SNAP_TIMEZONE | INTERVAL DAY(0) TO SECOND(0) | | Snapshot time zone expressed as offset from UTC (Coordinated Universal Time) time zone |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"DBA_HIST_SNAPSHOT"

# DBA_HIST_BASELINE

DBA_HIST_BASELINE displays information on baselines taken in the system. For each baseline, this view displays the complete time range and whether the baseline is the default baseline.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | | Database ID |
| BASELINE_ID | NUMBER | | Internal ID for the baseline |
| BASELINE_NAME | VARCHAR2(64) | | User-specified name for the baseline |
| BASELINE_TYPE | VARCHAR2(13) | | The baseline type, as follows:<br>STATIC - baselines that are created manually by the user<br>MOVING WINDOW - baselines that have dynamic start and end snapshot IDs<br>GENERATED - baselines that are automatically generated by the system, using a template |
| START_SNAP_ID | NUMBER | | Start snapshot ID for the baseline |
| START_SNAP_TIME | TIMESTAMP(3) | | Time associated with the start snapshot ID |
| END_SNAP_ID | NUMBER | | End snapshot ID for the baseline |
| END_SNAP_TIME | TIMESTAMP(3) | | Time associated with the end snapshot ID |
| MOVING_WINDOW_SIZE | NUMBER | | If BASELINE_TYPE is MOVING WINDOW, this field is the size of the moving window in number of days. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | If NULL, then the window size is the value of the AWR retention setting. |
| CREATION_TIME | DATE | | Time the baseline was created |
| EXPIRATION | NUMBER | | How long to keep the baseline, in number of days. A NULL value means that the baseline will be kept forever. |
| TEMPLATE_NAME | VARCHAR2(64) | | Name of the template that created this baseline, if any. |
| LAST_TIME_COMPUTED | DATE | | Last time that statistics were computed on the baseline. |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_BASELINE_DETAILS

DBA_HIST_BASELINE_DETAILS displays details about the baseline.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | | Database ID |
| INSTANCE_NUMBER | NUMBER | | Instance ID for the baseline data |
| BASELINE_ID | NUMBER | | Internal ID for the baseline |
| BASELINE_NAME | VARCHAR2(64) | | User-specified name for the baseline |
| BASELINE_TYPE | VARCHAR2(13) | | The baseline type, as follows:<br>STATIC - baselines that are created manually by the user<br>MOVING WINDOW - baselines that have dynamic start and end snapshot IDs<br>GENERATED - baselines that are automatically generated by the system, using a template |
| START_SNAP_ID | NUMBER | | Start snapshot ID for the baseline |
| START_SNAP_TIME | TIMESTAMP(3) | | Start snapshot time for the baseline |
| END_SNAP_ID | NUMBER | | End snapshot ID for the baseline |
| END_SNAP_TIME | TIMESTAMP(3) | | End snapshot time for the baseline |
| SHUTDOWN | VARCHAR2(3) | | Whether or not there is a database startup or shutdown in this interval (YES, NO, or NULL). |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ERROR_COUNT | NUMBER | | Number of errors in the snapshots in the baseline snapshot range |
| PCT_TOTAL_TIME | NUMBER | | Amount of time captured in snapshots, divided by the total possible time for this baseline |
| LAST_TIME_COMPUTED | DATE | | Last time that statistics were computed on the baseline |
| MOVING_WINDOW_SIZE | NUMBER | | If BASELINE_TYPE is MOVING WINDOW, this field is the size of the moving window in number of days.<br>If NULL, then the window size is the value of the AWR retention setting. |
| CREATION_TIME | DATE | | Time the baseline was created |
| EXPIRATION | NUMBER | | How long to keep the baseline, in number of days. A value of NULL indicates that the baseline will be kept forever. |
| TEMPLATE_NAME | VARCHAR2(64) | | Name of the template that created this baseline, if any. |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_BASELINE_METADATA

DBA_HIST_BASELINE_METADATA displays metadata information for the baseline.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| BASELINE_ID | NUMBER | NOT NULL | Internal ID for the baseline |
| BASELINE_NAME | VARCHAR2(64) | | User-specified name for the baseline |
| BASELINE_TYPE | VARCHAR2(13) | | The baseline type, as follows:<br>STATIC - baselines that are created manually by the user<br>MOVING WINDOW - baselines that have dynamic start and end snapshot IDs<br>GENERATED - baselines that are automatically generated by the system, using a template |
| START_SNAP_ID | NUMBER | | Start snapshot ID for the baseline |
| END_SNAP_ID | NUMBER | | End snapshot ID for the baseline |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MOVING_WINDOW_SIZE | NUMBER | | If BASELINE_TYPE is MOVING WINDOW, this field is the size of the moving window in number of days.<br>If NULL, then the window size is the value of the AWR retention setting. |
| CREATION_TIME | DATE | | Time the baseline was created |
| EXPIRATION | NUMBER | | How long to keep the baseline, in number of days. If the value is NULL, the baseline will be kept forever. |
| TEMPLATE_NAME | VARCHAR2(64) | | Name of the template that created this baseline, if any |
| LAST_TIME_COMPUTED | DATE | | Last time that statistics were computed on the baseline |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_BASELINE_TEMPLATE

DBA_HIST_BASELINE_TEMPLATE displays the templates used by the system for baseline generation. The system uses this information to determine which baselines should be automatically created or removed.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| TEMPLATE_ID | NUMBER | NOT NULL | Internal ID for the template |
| TEMPLATE_NAME | VARCHAR2(30) | NOT NULL | Name of the template |
| TEMPLATE_TYPE | VARCHAR2(9) | NOT NULL | Type of the template, as follows:<br>SINGLE - one time period<br>REPEATING - maintain a time period |
| BASELINE_NAME_PREFIX | VARCHAR2(30) | NOT NULL | Name to use for the baselines that are created:<br>For a template type of SINGLE, the BASELINE_NAME_PREFIX is the name that will be used.<br>For a template type of REPEATING, the BASELINE_NAME will be the prefix to the name. |
| START_TIME | DATE | NOT NULL | For a template type of SINGLE, this is the start time for future baselines |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | For a template type of REPEATING, this is the effective start time that baselines should start being generated. |
| END_TIME | DATE | NOT NULL | For a template type of SINGLE, this is the end time for future baselines.<br>For a template type of REPEATING, this is the effective end time that baselines should stop being generated. |
| DAY_OF_WEEK | VARCHAR2(9) | | For a template type of REPEATING, this indicates the day of the week to create the baseline: SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, ALL. |
| HOUR_IN_DAY | NUMBER | | For a template type of REPEATING, a value from 0 - 23 to indicate the hour of the day to create the baseline for. |
| DURATION | NUMBER | | For a template type of REPEATING, the length of time for the baseline to be created. |
| EXPIRATION | NUMBER | | How long to keep the baseline, in number of days |
| REPEAT_INTERVAL | VARCHAR2(128) | | String that represents the time repeating information in the format used by the DBMS_SCHEDULER package |
| LAST_GENERATED | DATE | | Last time a baseline was generated for this template |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SCHEDULER package

# DBA_HIST_BG_EVENT_SUMMARY

DBA_HIST_BG_EVENT_SUMMARY displays the historical summary background event activity. This view contains snapshots from V$SESSION_EVENT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| EVENT_ID | NUMBER | NOT NULL | Identifier of the wait event |
| EVENT_NAME | VARCHAR2(64) | NOT NULL | Name of the wait event |
| WAIT_CLASS_ID | NUMBER | | Identifier of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | | Name of the class of the wait event |
| TOTAL_WAITS | NUMBER | | Total number of waits for the event |
| TOTAL_TIMEOUTS | NUMBER | | Total number of timeouts for the event |
| TIME_WAITED_MICRO | NUMBER | | Total amount of time waited for the event (in microseconds) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SESSION_EVENT"

# DBA_HIST_BUFFER_POOL_STAT

DBA_HIST_BUFFER_POOL_STAT displays historical statistics about all buffer pools available for the instance. This view contains snapshots of V$BUFFER_POOL_STATISTICS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| ID | NUMBER | NOT NULL | Buffer pool identifier number |
| NAME | VARCHAR2(20) | | Name of the buffer pool |
| BLOCK_SIZE | NUMBER | | Block Size |
| SET_MSIZE | NUMBER | | Buffer pool maximum set size |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CNUM_REPL | NUMBER | | Number of buffers on the replacement list |
| CNUM_WRITE | NUMBER | | Number of buffers on the write list |
| CNUM_SET | NUMBER | | Number of buffers in the set |
| BUF_GOT | NUMBER | | Number of buffers gotten by the set |
| SUM_WRITE | NUMBER | | Number of buffers written by the set |
| SUM_SCAN | NUMBER | | Number of buffers scanned in the set |
| FREE_BUFFER_WAIT | NUMBER | | Free buffer wait statistic |
| WRITE_COMPLETE_WAIT | NUMBER | | Write complete wait statistic |
| BUFFER_BUSY_WAIT | NUMBER | | Buffer busy wait statistic |
| FREE_BUFFER_INSPECTED | NUMBER | | Free buffer inspected statistic |
| DIRTY_BUFFERS_INSPECTED | NUMBER | | Dirty buffers inspected statistic |
| DB_BLOCK_CHANGE | NUMBER | | Database blocks changed statistic |
| DB_BLOCK_GETS | NUMBER | | Database blocks gotten statistic |
| CONSISTENT_GETS | NUMBER | | Consistent gets statistic |
| PHYSICAL_READS | NUMBER | | Physical reads statistic |
| PHYSICAL_WRITES | NUMBER | | Physical writes statistic |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$BUFFER_POOL_STATISTICS"

# DBA_HIST_BUFFERED_QUEUES

DBA_HIST_BUFFERED_QUEUES displays historical information about all buffered queues available for the instance.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| QUEUE_SCHEMA | VARCHAR2(128) | NOT NULL | Owner of the queue |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| STARTUP_TIME | DATE | NOT NULL | Startup time of the instance |
| QUEUE_ID | NUMBER | NOT NULL | ID of the queue |
| NUM_MSGS | NUMBER | | Total number of outstanding messages currently enqueued in the buffered queue for the subscriber (includes the count of the messages overflowed to disk) |
| SPILL_MSGS | NUMBER | | Current number of overflow messages spilled to disk from the buffered queue |
| CNUM_MSGS | NUMBER | | Cumulative total number of messages enqueued into the buffered queue since the buffered queue was created. |
| CSPILL_MSGS | NUMBER | | Cumulative total number of overflow messages spilled to disk from the buffered queue since the buffered queue was created |
| EXPIRED_MSGS | NUMBER | | Number of expired messages |
| OLDEST_MSGID | RAW(16) | | Message ID of the oldest message |
| OLDEST_MSG_ENQTM | TIMESTAMP(3) | | Enqueue time of the oldest message |
| QUEUE_STATE | VARCHAR2(25) | | Indicates whether the queue is in recovery mode (QUEUE IS IN RECOVERY MODE) or not (NORMAL) |
| ELAPSED_ENQUEUE_TIME | NUMBER | | Total time spent in enqueue (in hundredths of a second) |
| ELAPSED_DEQUEUE_TIME | NUMBER | | Total time spent in dequeue (in hundredths of a second) |
| ELAPSED_TRANSFORMATION_TIME | NUMBER | | Total time for evaluating transformations (in hundredths of a second) |
| ELAPSED_RULE_EVALUATION_TIME | NUMBER | | Total time for rule evaluations (in hundredths of a second) |
| ENQUEUE_CPU_TIME | NUMBER | | Total CPU time for enqueue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | | Total CPU time for dequeue (in hundredths of a second) |
| LAST_ENQUEUE_TIME | TIMESTAMP(3) | | Last message enqueue time |
| LAST_DEQUEUE_TIME | TIMESTAMP(3) | | Last message dequeue time |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_BUFFERED_SUBSCRIBERS

DBA_HIST_BUFFERED_SUBSCRIBERS displays historical information about the subscribers for all buffered queues in the instance.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| QUEUE_SCHEMA | VARCHAR2(128) | NOT NULL | Owner of the queue |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| SUBSCRIBER_ID | NUMBER | NOT NULL | Internal subscriber number (for identification) |
| SUBSCRIBER_NAME | VARCHAR2(128) | | Name of the subscriber |
| SUBSCRIBER_ADDRESS | VARCHAR2(1024) | | Address of the subscribing agent |
| SUBSCRIBER_TYPE | VARCHAR2(128) | | Type of subscriber:<br>• PROXY - Proxy subscriber<br>• SUBSCRIBOR |
| STARTUP_TIME | DATE | NOT NULL | Startup time of the instance |
| LAST_BROWSED_SEQ | NUMBER | | Sequence number of the most recently browsed message for the subscriber (comparable to the number of messages in the V$STREAMS_APPLY_READER view) |
| LAST_BROWSED_NUM | NUMBER | | Internal Message number for the most recently browsed message for the subscriber |
| LAST_DEQUEUED_SEQ | NUMBER | | Sequence number of the most recently dequeued message for the subscriber (comparable to the number of messages in the V$STREAMS_APPLY_COORDINATOR view) |
| LAST_DEQUEUED_NUM | NUMBER | | Internal Message number for the most recently dequeued message for the subscriber |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CURRENT_ENQ_SEQ | NUMBER | | Current sequence number of the most recently enqueued message for the subscriber |
| NUM_MSGS | NUMBER | | Total number of outstanding messages currently enqueued in the buffered queue for the subscriber (includes the count of the messages overflowed to disk) |
| CNUM_MSGS | NUMBER | | Cumulative total number of messages enqueued for the subscriber since the creation of the buffered queue |
| TOTAL_DEQUEUED_MSG | NUMBER | | Total number of messages dequeued by the subscriber |
| TOTAL_SPILLED_MSG | NUMBER | | Total number of spilled messages for the subscriber |
| EXPIRED_MSGS | NUMBER | | Number of expired messages |
| MESSAGE_LAG | NUMBER | | Message lag of the subscriber |
| ELAPSED_DEQUEUE_TIME | NUMBER | | Total time spent in dequeue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | | Total CPU time for dequeue (in hundredths of a second) |
| LAST_DEQUEUE_TIME | TIMESTAMP(3) | | Last message dequeue time |
| OLDEST_MSGID | RAW(16) | | Message ID of the oldest message |
| OLDEST_MSG_ENQTM | TIMESTAMP(3) | | Enqueue time of the oldest message |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_CAPTURE

DBA_HIST_CAPTURE displays historical statistics information about each capture process for Oracle Streams, Oracle GoldenGate, and XStream capture operations. This view is intended for use with Automatic Workload Repository (AWR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the capture process |
| STARTUP_TIME | DATE | NOT NULL | Time that the capture process was last started |
| LAG | NUMBER | | Delay (in seconds) between the creation and capture of the most recently captured message |
| TOTAL_MESSAGES_CAPTURED | NUMBER | | Total changes captured since the capture process was last started |
| TOTAL_MESSAGES_ENQUEUED | NUMBER | | Total number of messages enqueued since the capture process was last started |
| ELAPSED_RULE_TIME | NUMBER | | Elapsed time (in hundredths of a second) evaluating rules since the capture process was last started |
| ELAPSED_ENQUEUE_TIME | NUMBER | | Elapsed time (in hundredths of a second) enqueuing messages since the capture process was last started |
| ELAPSED_REDO_WAIT_TIME | NUMBER | | Elapsed time (in hundredths of a second) spent by the capture process in the WAITING FOR REDO state since the capture process was last started. |
| ELAPSED_PAUSE_TIME | NUMBER | | Elapsed pause time (in hundredths of a second) spent by the capture process since the capture process was last restarted |
| CON_DBID | NUMBER | | The database ID of the PDB |
| EXTRACT_NAME | VARCHAR2(128) | | Name of the extract process, if applicable |
| BYTES_REDO_MINED | NUMBER | | The total amount of redo data mined (in bytes) since the capture process last started |
| BYTES_SENT | NUMBER | | Total number of bytes sent by the capture process to the extract process since the last time the extract process attached to the capture process |
| SESSION_MODULE | VARCHAR2(64) | NOT NULL | Session module. Valid values:<br>• GoldenGate<br>• Streams<br>• XStream |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_CLUSTER_INTERCON

DBA_HIST_CLUSTER_INTERCON displays information about the devices used by the instance to access the interconnect (that is, communicate with other instances).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| NAME | VARCHAR2(256) | NOT NULL | Operating system name of the device |
| IP_ADDRESS | VARCHAR2(64) | NOT NULL | IP address of the device |
| IS_PUBLIC | VARCHAR2(3) | | Indicates whether the device is a public interface (YES) or a private interface (NO)<br>Public interfaces can be listened to by outside applications, which may be a security problem. Oracle recommends using private interfaces for interconnect. |
| SOURCE | VARCHAR2(31) | | Describes the type of device |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_COLORED_SQL

DBA_HIST_COLORED_SQL displays the SQL IDs that have been marked for AWR SQL capture. If a SQL statement is colored using the DBMS_WORKLOAD_REPOSITORY.ADD_COLORED_SQL procedure, then AWR will always capture the SQL statistics for the colored SQL ID. A SQL statement can be removed from coloring using the DBMS_WORKLOAD_REPOSITORY.REMOVE_COLORED_SQL procedure.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL ID of colored SQL statement |
| CREATE_TIME | DATE | NOT NULL | Time the SQL ID was colored |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_WORKLOAD_REPOSITORY package.

# DBA_HIST_COMP_IOSTAT

DBA_HIST_COMP_IOSTAT displays information about I/O statistics aggregated on the component level.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| COMPONENT | VARCHAR2(64) | NOT NULL | Component name |
| FILE_TYPE | VARCHAR2(64) | NOT NULL | File type |
| IO_TYPE | CHAR(5) | NOT NULL | The type of I/O performed |
| OPERATION | CHAR(5) | NOT NULL | Operation name |
| BYTES | NUMBER | NOT NULL | Number of bytes |
| IO_COUNT | NUMBER | NOT NULL | Number of I/Os that were performed |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_CR_BLOCK_SERVER

DBA_HIST_CR_BLOCK_SERVER displays historical statistics on the Global Cache Service processes (lMS) used in cache fusion. This view contains snapshots of V $CR_BLOCK_SERVER.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| CR_REQUESTS | NUMBER | | Number of CR blocks served due to remote CR block requests |
| CURRENT_REQUESTS | NUMBER | | Number of current blocks served due to remote CR block requests<br>CR_REQUESTS + CURRENT_REQUESTS = global cache CR blocks served (from V$SYSSTAT). |
| DATA_REQUESTS | NUMBER | | Number of current or CR requests for data blocks |
| UNDO_REQUESTS | NUMBER | | Number of CR requests for undo blocks |
| TX_REQUESTS | NUMBER | | Number of CR requests for undo segment header blocks<br>DATA_REQUESTS + UNDO_REQUESTS + TX_REQUESTS = total number of requests handled by the LMS processes |
| CURRENT_RESULTS | NUMBER | | Number of requests for which no changes were rolled out of the block returned to the requesting instance |
| PRIVATE_RESULTS | NUMBER | | Number of requests for which changes were rolled out of the block returned to the requesting instance, and only the requesting transaction can use the resulting CR block |
| ZERO_RESULTS | NUMBER | | Number of requests for which changes were rolled out of the block returned to the requesting instance. Only zero-XID transactions can use the block. |
| DISK_READ_RESULTS | NUMBER | | Number of requests for which the requesting instance had to read the requested block from disk |
| FAIL_RESULTS | NUMBER | | Number of requests that failed; the requesting transaction must reissue the request |
| FAIRNESS_DOWN_CONVERTS | NUMBER | | Number of times an instance receiving a request has down-converted an X lock on a block because it was not modifying the block |
| FAIRNESS_CLEARS | NUMBER | | Number of times the fairness counter was cleared. This counter tracks the number of times a block was modified after it was served. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FREE_GC_ELEMENTS | NUMBER | | Number of times a request was received from another instance and the X lock had no buffers |
| FLUSHES | NUMBER | | Number of times the log has been flushed by an LMS process |
| FLUSHES_QUEUED | NUMBER | | Number of flushes queued by an LMS process |
| FLUSH_QUEUE_FULL | NUMBER | | Number of times the flush queue was full |
| FLUSH_MAX_TIME | NUMBER | | Maximum time for flush |
| LIGHT_WORKS | NUMBER | | Number of times the light-work rule was evoked. This rule prevents the LMS processes from going to disk while responding to CR requests for data, undo, or undo segment header blocks. This rule can prevent the LMS process from completing its response to the CR request. |
| ERRORS | NUMBER | | Number of times an error was signalled by an LMS process |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$CR_BLOCK_SERVER"

# DBA_HIST_CURRENT_BLOCK_SERVER

DBA_HIST_CURRENT_BLOCK_SERVER displays historical statistics on the Global Cache Service processes (lMS) used in cache fusion. This view contains snapshots of V$CURRENT_BLOCK_SERVER.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| PIN1 | NUMBER | | Pins taking less than 1 millisecond |
| PIN10 | NUMBER | | Pins taking 1 to 10 milliseconds |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PIN100 | NUMBER | | Pins taking 10 to 100 milliseconds |
| PIN1000 | NUMBER | | Pins taking 100 to 1000 milliseconds |
| PIN10000 | NUMBER | | Pins taking 1000 to 10000 milliseconds |
| FLUSH1 | NUMBER | | Flushes taking less than 1 millisecond |
| FLUSH10 | NUMBER | | Flushes taking 1 to 10 milliseconds |
| FLUSH100 | NUMBER | | Flushes taking 10 to 100 milliseconds |
| FLUSH1000 | NUMBER | | Flushes taking 100 to 1000 milliseconds |
| FLUSH10000 | NUMBER | | Flushes taking 1000 to 10000 milliseconds |
| WRITE1 | NUMBER | | Writes taking less than 1 millisecond |
| WRITE10 | NUMBER | | Writes taking 1 to 10 milliseconds |
| WRITE100 | NUMBER | | Writes taking 10 to 100 milliseconds |
| WRITE1000 | NUMBER | | Writes taking 100 to 1000 milliseconds |
| WRITE10000 | NUMBER | | Writes taking 1000 to 10000 milliseconds |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$CURRENT_BLOCK_SERVER"

# DBA_HIST_DATABASE_INSTANCE

DBA_HIST_DATABASE_INSTANCE displays the databases and instances in the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number |
| STARTUP_TIME | TIMESTAMP(3) | NOT NULL | Startup time of the instance |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARALLEL | VARCHAR2(3) | NOT NULL | Indicates whether the instance is running in an Oracle Real Application Clusters (Oracle RAC) environment (YES) or not (NO) |
| VERSION | VARCHAR2(17) | NOT NULL | Database version |
| DB_NAME | VARCHAR2(9) | | Name of the database |
| INSTANCE_NAME | VARCHAR2(16) | | Name of the instance |
| HOST_NAME | VARCHAR2(64) | | Name of the host |
| LAST_ASH_SAMPLE_ID | NUMBER | NOT NULL | Last sample ID for the active session history |
| PLATFORM_NAME | VARCHAR2(101) | | Platform on which the instance is running |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_DATAFILE

DBA_HIST_DATAFILE displays a history of the data file information from the control file. This view contains snapshots of V$DATAFILE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| FILE# | NUMBER | NOT NULL | File identification number |
| CREATION_CHANGE# | NUMBER | NOT NULL | Change number at which the data file was created |
| FILENAME | VARCHAR2(513) | NOT NULL | Name of the data file |
| TS# | NUMBER | NOT NULL | Tablespace number |
| TSNAME | VARCHAR2(30) | | Name of the tablespace |
| BLOCK_SIZE | NUMBER | | Block size of the data file |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$DATAFILE"

# DBA_HIST_DB_CACHE_ADVICE

DBA_HIST_DB_CACHE_ADVICE displays historical predictions of the number of physical reads for the cache size corresponding to each row. This view contains snapshots of V$DB_CACHE_ADVICE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| BPID | NUMBER | NOT NULL | Buffer Pool identifier (ranges from 1 to 8) |
| BUFFERS_FOR_ESTIMATE | NUMBER | NOT NULL | Cache size for prediction (in terms of buffers) |
| NAME | VARCHAR2(20) | | Buffer pool name |
| BLOCK_SIZE | NUMBER | | Block size in bytes for buffers in the pool (the standard block size, the power of 2 nonstandard block sizes, 2048, 4096, 8192, 16384, or 32768) |
| ADVICE_STATUS | VARCHAR2(3) | | Status of the advisory:<br>• ON - Currently running<br>• OFF - Disabled (the estimates are historical and calculated when last enabled) |
| SIZE_FOR_ESTIMATE | NUMBER | | Cache size for prediction (in megabytes) |
| SIZE_FACTOR | NUMBER | | Size factor with respect to the current cache size |
| PHYSICAL_READS | NUMBER | | Physical reads for the cache size |
| BASE_PHYSICAL_READS | NUMBER | | Base physical reads for the cache size |
| ACTUAL_PHYSICAL_READS | NUMBER | | Actual physical reads for the cache size |
| ESTD_PHYSICAL_READ_TIME | NUMBER | | Estimated physical read time for the cache size |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$DB_CACHE_ADVICE"

# DBA_HIST_DISPATCHER

DBA_HIST_DISPATCHER displays historical information for each dispatcher process present at the time of the snapshot. This view contains snapshots of information from V$DISPATCHER and V$QUEUE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| NAME | VARCHAR2(4) | NOT NULL | Name of the dispatcher process |
| SERIAL# | NUMBER | | Serial number of the dispatcher process |
| IDLE | NUMBER | | Total idle time for the dispatcher (in hundredths of a second) |
| BUSY | NUMBER | | Total busy time for the dispatcher (in hundredths of a second) |
| WAIT | NUMBER | | Total time that all items in the dispatcher queue have waited (in hundredths of a second). Divide by TOTALQ for average wait per item. |
| TOTALQ | NUMBER | | Total number of items that have ever been in the dispatcher queue |
| SAMPLED_TOTAL_CONN | NUMBER | | Cumulative sum of total number of connections to the dispatcher over all samples. To determine the average number of connections to the dispatcher between two snapshots, divide the difference in SAMPLED_TOTAL_CONN by the difference in NUM_SAMPLES (obtained from DBA_HIST_SHARED_SERVER_SUMMARY). |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$DISPATCHER"
- "V$QUEUE"

# DBA_HIST_DLM_MISC

DBA_HIST_DLM_MISC displays miscellaneous Oracle Real Application Clusters (Oracle RAC) statistics. This view contains snapshots of V$DLM_MISC.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | | Unique snapshot ID |
| DBID | NUMBER | | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | | Instance number for the snapshot |
| STATISTIC# | NUMBER | | Statistic number |
| NAME | VARCHAR2(38) | | Statistic name |
| VALUE | NUMBER | | Statistic value |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_DYN_REMASTER_STATS

DBA_HIST_DYN_REMASTER_STATS displays historical statistical information about the dynamic remastering process. All times are given in hundredths of a second, and total values reflect what has been collected since instance startup. This view contains snapshots of V$DYNAMIC_REMASTER_STATS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| REMASTER_TYPE[1] | VARCHAR2(11) | NOT NULL | Remaster process type. Possible values:<br>• AFFINITY: This value is used for the row containing statistics that pertain to dynamic remastering activity on object affinity.<br>• READ-MOSTLY: This value is used for the row containing statistics that pertain to dynamic remastering activity on read-mostly objects. |
| REMASTER_OPS | NUMBER | | Total number of dynamic remastering operations |
| REMASTER_TIME | NUMBER | | Total dynamic remastering time |
| REMASTERED_OBJECTS | NUMBER | | Total number of objects dynamically remastered due to affinity |
| QUIESCE_TIME | NUMBER | | Total quiesce step time |
| FREEZE_TIME | NUMBER | | Total freeze step time |
| CLEANUP_TIME | NUMBER | | Total cleanup step time |
| REPLAY_TIME | NUMBER | | Total replay step time |
| FIXWRITE_TIME | NUMBER | | Total fixwrite step time |
| SYNC_TIME | NUMBER | | Total synchronization step time |
| RESOURCES_CLEANED | NUMBER | | Total number of resources cleaned in the cleanup steps |
| REPLAYED_LOCKS_SENT | NUMBER | | Total number of locks replayed to other instances in the replay steps |
| REPLAYED_LOCKS_RECEIVED | NUMBER | | Total number of locks received from other instances in the replay steps |
| CURRENT_OBJECTS | NUMBER | | Current number of objects remastered on this instance due to affinity |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"V$DYNAMIC_REMASTER_STATS"

# DBA_HIST_ENQUEUE_STAT

DBA_HIST_ENQUEUE_STAT displays historical statistics on the number of enqueue (lock) requests for each type of lock. This view contains snapshots of V$ENQUEUE_STATISTICS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| EQ_TYPE | VARCHAR2(2) | NOT NULL | Type of enqueue requested |
| REQ_REASON | VARCHAR2(64) | NOT NULL | Reason for the enqueue request |
| TOTAL_REQ# | NUMBER | | Total number of enqueue requests or enqueue conversions for this type of enqueue |
| TOTAL_WAIT# | NUMBER | | Total number of times an enqueue request or conversion resulted in a wait |
| SUCC_REQ# | NUMBER | | Number of times an enqueue request or conversion was granted |
| FAILED_REQ# | NUMBER | | Number of times an enqueue request or conversion failed |
| CUM_WAIT_TIME | NUMBER | | Total amount of time (in milliseconds) spent waiting for the enqueue or enqueue conversion |
| EVENT# | NUMBER | | Event number |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$ENQUEUE_STATISTICS"

# DBA_HIST_EVENT_HISTOGRAM

DBA_HIST_EVENT_HISTOGRAM displays event histogram historical statistics information. This view contains snapshots of V$EVENT_HISTOGRAM.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| EVENT_ID | NUMBER | NOT NULL | Identifier of the wait event |
| EVENT_NAME | VARCHAR2(64) | NOT NULL | Name of the wait event |
| WAIT_CLASS_ID | NUMBER | | Identifier of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | | Name of the class of the wait event |
| WAIT_TIME_MILLI | NUMBER | NOT NULL | Wait time (in milliseconds) |
| WAIT_COUNT | NUMBER | | Wait count |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$EVENT_HISTOGRAM"

# DBA_HIST_EVENT_NAME

DBA_HIST_EVENT_NAME displays information about wait events. This view contains a snapshot of V$EVENT_NAME.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| EVENT_ID | NUMBER | NOT NULL | Identifier of the wait event |
| EVENT_NAME | VARCHAR2(64) | NOT NULL | Name of the wait event |
| PARAMETER1 | VARCHAR2(64) | | Description of the first parameter for the wait event |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETER2 | VARCHAR2(64) | | Description of the second parameter for the wait event |
| PARAMETER3 | VARCHAR2(64) | | Description of the third parameter for the wait event |
| WAIT_CLASS_ID | NUMBER | | Identifier of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | | Name of the class of the wait event |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$EVENT_NAME"

# DBA_HIST_FILEMETRIC_HISTORY

DBA_HIST_FILEMETRIC_HISTORY displays the history of file metrics collected in the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| FILEID | NUMBER | NOT NULL | File number |
| CREATIONTIME | NUMBER | NOT NULL | File creation time |
| BEGIN_TIME | DATE | NOT NULL | Begin time of the interval |
| END_TIME | DATE | NOT NULL | End time of the interval |
| INTSIZE | NUMBER | NOT NULL | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | NOT NULL | ID of the group to which the file belongs |
| AVGREADTIME | NUMBER | NOT NULL | Average file read time |
| AVGWRITETIME | NUMBER | NOT NULL | Average file write time |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PHYSICALREAD | NUMBER | NOT NULL | Number of physical reads |
| PHYSICALWRITE | NUMBER | NOT NULL | Number of physical writes |
| PHYBLKREAD | NUMBER | NOT NULL | Number of physical block reads |
| PHYBLKWRITE | NUMBER | NOT NULL | Number of physical block writes |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_FILESTATXS

DBA_HIST_FILESTATXS displays information about file read/write statistics. This view contains snapshots of V$FILESTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| FILE# | NUMBER | NOT NULL | File identification number |
| CREATION_CHANGE# | NUMBER | NOT NULL | Change number at which the data file was created |
| FILENAME | VARCHAR2(513) | NOT NULL | Name of the data file |
| TS# | NUMBER | NOT NULL | Tablespace number |
| TSNAME | VARCHAR2(30) | | Name of the tablespace |
| BLOCK_SIZE | NUMBER | | Block size of the data file |
| PHYRDS | NUMBER | | Number of physical reads done |
| PHYWRTS | NUMBER | | Number of times DBWR is required to write |
| SINGLEBLKRDS | NUMBER | | Number of single block reads |
| READTIM | NUMBER | | Time (in hundredths of a second) spent doing reads if the TIMED_STATISTICS parameter is true; 0 if TIMED_STATISTICS is false |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WRITETIM | NUMBER | | Time (in hundredths of a second) spent doing writes if the TIMED_STATISTICS parameter is true; 0 if TIMED_STATISTICS is false |
| SINGLEBLKRDTIM | NUMBER | | Cumulative single block read time (in hundredths of a second) |
| PHYBLKRD | NUMBER | | Number of physical blocks read |
| PHYBLKWRT | NUMBER | | Number of blocks written to disk, which may be the same as PHYWRTS if all writes are single blocks |
| WAIT_COUNT | NUMBER | | Shows the number of waits at the file level for contended buffers. This value includes the individual wait events that are included in the buffer busy waits wait event.<br>**See Also:** "buffer busy waits" |
| TIME | NUMBER | | Time spent waiting for the wait events in the WAIT_COUNT column |
| OPTIMIZED_PHYBLKRD | NUMBER | | Number of physical reads from Database Smart Flash Cache blocks |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$FILESTAT"

# DBA_HIST_IC_CLIENT_STATS

DBA_HIST_IC_CLIENT_STATS displays information about the usage of an interconnect device by the instance. The information is divided into several areas of the Oracle Database, each identified by the NAME value.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(9) | NOT NULL | Identifies the area of the Oracle Database:<br>• ipq - Parallel query communications<br>• dlm - Database lock management<br>• cache - Global cache communications<br>All other values are internal to Oracle and are not expected to have high usage. |
| BYTES_SENT | NUMBER | | Number of bytes sent by the instance since instance startup for the software area identified by NAME. This information is aggregated across all devices used by the instance. |
| BYTES_RECEIVED | NUMBER | | Number of bytes received by the instance since instance startup for the software area identified by NAME. This information is aggregated across all devices used by the instance. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_IC_DEVICE_STATS

DBA_HIST_IC_DEVICE_STATS displays operating system information about the usage of interconnect devices by the machine. This usage contains Oracle usage but is not limited to it. The quality of the information depends on the operating system.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| IF_NAME | VARCHAR2(256) | NOT NULL | Name of the device (same as NAME in DBA_HIST_CLUSTER_INTERCON) |
| IP_ADDR | VARCHAR2(64) | NOT NULL | IP address of the device (same as IP_ADDRESS in DBA_HIST_CLUSTER_INTERCON) |
| NET_MASK | VARCHAR2(16) | | Network mask |
| FLAGS | VARCHAR2(32) | | Flags |
| MTU | NUMBER | | Maximum transmission unit |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BYTES_RECEIVED | NUMBER | | Number of bytes received since operating system start time |
| PACKETS_RECEIVED | NUMBER | | Number of packets received since operating system start time |
| RECEIVE_ERRORS | NUMBER | | Number of receive errors since operating system start time |
| RECEIVE_DROPPED | NUMBER | | Number of receive messages that were dropped |
| RECEIVE_BUF_OR | NUMBER | | Number of receive buffer overruns experienced |
| RECEIVE_FRAME_ERR | NUMBER | | Number of receive errors due to frame error |
| BYTES_SENT | NUMBER | | Number of bytes sent since operating system start time |
| PACKETS_SENT | NUMBER | | Number of packets sent since operating system start time |
| SEND_ERRORS | NUMBER | | Number of send errors since operating system start time |
| SENDS_DROPPED | NUMBER | | Number of send messages that were dropped |
| SEND_BUF_OR | NUMBER | | Number of send buffer overruns experienced |
| SEND_CARRIER_LOST | NUMBER | | Number of send errors due to carrier lost |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_IM_SEG_STAT

DBA_HIST_IM_SEG_STAT displays information about historical in-memory segment statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| TS# | NUMBER | NOT NULL | Tablespace number |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJ# | NUMBER | NOT NULL | Dictionary object number |
| DATAOBJ# | NUMBER | NOT NULL | Data object number |
| MEMBYTES | NUMBER | | Size of in-memory version of the segment in bytes |
| SCANS | NUMBER | | Count of segment statistics |
| SCANS_DELTA | NUMBER | | Delta values for in-memory scans |
| DB_BLOCK_CHANGES | NUMBER | | The total number of changes that were part of an update or delete operation that were made to segment blocks |
| DB_BLOCK_CHANGES_DELTA | NUMBER | | Delta value for database block changes |
| POPULATE_CUS | NUMBER | | Count of compression units (CUs) populated per segment |
| POPULATE_CUS_DELTA | NUMBER | | Delta value for compression unit (CU) populate operations |
| REPOPULATE_CUS | NUMBER | | Count of CUs repopulated per segment |
| REPOPULATE_CUS_DELTA | NUMBER | | Delta value for compression unit (CU) repopulate operations |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# DBA_HIST_IM_SEG_STAT_OBJ

DBA_HIST_IM_SEG_STAT_OBJ displays information about object metadata for historical in-memory segments.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database id for the snapshot |
| TS# | NUMBER | NOT NULL | Tablespace number |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJ# | NUMBER | NOT NULL | Dictionary object number |
| DATAOBJ# | NUMBER | NOT NULL | Data object number |
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject |
| OBJECT_TYPE | VARCHAR2(128) | | Type of the object |
| TABLESPACE_NAME | VARCHAR2(128) | | Tablespace name for the object |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# DBA_HIST_INST_CACHE_TRANSFER

DBA_HIST_INST_CACHE_TRANSFER displays the historical statistics on the cache blocks transferred among instances. This view contains snapshots of V $INSTANCE_CACHE_TRANSFER.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| INSTANCE | NUMBER | NOT NULL | Instance from which the blocks are transferred |
| CLASS | VARCHAR2(18) | NOT NULL | Class of the cache block |
| CR_BLOCK | NUMBER | | CR block transfers not affected by remote processing delays |
| CR_BUSY | NUMBER | | Current block transfers affected by remote contention |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CR_CONGESTED | NUMBER | | CR block transfers affected by remote system load |
| CURRENT_BLOCK | NUMBER | | Current block transfers not affected by remote processing delays |
| CURRENT_BUSY | NUMBER | | Current block transfers affected by remote contention |
| CURRENT_CONGESTED | NUMBER | | Current block transfers affected by remote system load |
| LOST | NUMBER | | The number of blocks that were sent by a particular instance but that never arrived in this instance |
| CR_2HOP | NUMBER | | The count of CR blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CR_3HOP | NUMBER | | The count of CR blocks which were received by this instance from a particular instance after a 3-way round-trip |
| CURRENT_2HOP | NUMBER | | The count of current blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CURRENT_3HOP | NUMBER | | The count of current blocks which were received by this instance from a particular instance after a 3-way round-trip |
| CR_BLOCK_TIME | NUMBER | | Total time waited for CR blocks from a particular instance (includes the other times) |
| CR_BUSY_TIME | NUMBER | | The time waited for CR blocks which were received by this instance from a particular instance and which were delayed by a log flushed on the sending instance |
| CR_CONGESTED_TIME | NUMBER | | The time waited for CR blocks which were received by this instance from a particular instance and which were delayed because LMS was busy |
| CURRENT_BLOCK_TIME | NUMBER | | Total time waited for CR blocks from a particular instance (includes the other times) |
| CURRENT_BUSY_TIME | NUMBER | | The time waited for current blocks which were received by this instance from a particular instance and which were delayed by a log flushed on the sending instance |
| CURRENT_CONGESTED_TIME | NUMBER | | The time waited for current blocks which were received by this instance from a particular instance and which were delayed because LMS was busy |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOST_TIME | NUMBER | | The time waited for blocks that were sent by a particular instance but that never arrived in this instance |
| CR_2HOP_TIME | NUMBER | | The time waited for CR blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CR_3HOP_TIME | NUMBER | | The time waited for CR blocks which were received by this instance from a particular instance after a 3-way round-trip |
| CURRENT_2HOP_TIME | NUMBER | | The time waited for current blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CURRENT_3HOP_TIME | NUMBER | | The time waited for current blocks which were received by this instance from a particular instance after a 3-way round-trip |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$INSTANCE_CACHE_TRANSFER"

# DBA_HIST_INSTANCE_RECOVERY

DBA_HIST_INSTANCE_RECOVERY displays the historical monitoring of the mechanisms available to the user to limit recovery I/O. This view contains snapshots of V$INSTANCE_RECOVERY.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| RECOVERY_ESTIMATED_IOS | NUMBER | | Number of dirty buffers in the buffer cache. In the Standard Edition, this column is always null. |
| ACTUAL_REDO_BLKS | NUMBER | | Current actual number of redo blocks required for recovery |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TARGET_REDO_BLKS | NUMBER | | Current target number of redo blocks that must be processed for recovery. This value is the minimum value of the following 3 columns, and identifies which of the 3 user-defined limits determines checkpointing. |
| LOG_FILE_SIZE_REDO_BLKS | NUMBER | | Maximum number of redo blocks required to guarantee that a log switch does not occur before the checkpoint completes |
| LOG_CHKPT_TIMEOUT_REDO_BLKS | NUMBER | | Number of redo blocks that need to be processed during recovery to satisfy the LOG_CHECKPOINT_TIMEOUT parameter. The value displayed is not meaningful unless LOG_CHECKPOINT_TIMEOUT has been set. |
| LOG_CHKPT_INTERVAL_REDO_BLKS | NUMBER | | Number of redo blocks that need to be processed during recovery to satisfy the LOG_CHECKPOINT_INTERVAL parameter. The value displayed is not meaningful unless LOG_CHECKPOINT_INTERVAL has been set. |
| FAST_START_IO_TARGET_REDO_BLKS | NUMBER | | This column is obsolete and maintained for backward compatibility. The value of this column is always null. |
| TARGET_MTTR | NUMBER | | Effective MTTR (mean time to recover) target value in seconds. The TARGET_MTTR value is calculated based on the value of the FAST_START_MTTR_TARGET parameter (the TARGET_MTTR value is used internally), and is usually an approximation of the parameter's value. However, if the FAST_START_MTTR_TARGET parameter value is very small (for example, one second), or very large (for example, 3600 seconds), then the calculation will produce a target value dictated by system limitations.<br>In such cases, the TARGET_MTTR value will be the shortest calculated time, or the longest calculated time that recovery is expected to take. If FAST_START_MTTR_TARGET is not specified, then the value of this field is the current estimated MTTR. |
| ESTIMATED_MTTR | NUMBER | | Current estimated mean time to recover (MTTR) based on the number of dirty buffers and log blocks (0 if FAST_START_MTTR_TARGET is not specified). This value tells you how long you can expect recovery to take based on the work the system is doing right now. |
| CKPT_BLOCK_WRITES | NUMBER | | Number of blocks written by checkpoint writes |
| OPTIMAL_LOGFILE_SIZE | NUMBER | | Redo log file size (in megabytes) that is considered optimal based on the current setting of FAST_START_MTTR_TARGET. It is recommended that all online redo logs be configured to be at least this value. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ESTD_CLUSTER_AVAILABLE_TIME | NUMBER | | Estimated time (in seconds) that the cluster would become partially available should the instance fail. This column is only meaningful in an Oracle Real Application Clusters (Oracle RAC) environment. In a non-Oracle RAC environment, the value of this column is null. |
| WRITES_MTTR | NUMBER | | Number of writes driven by the FAST_START_MTTR_TARGET parameter |
| WRITES_LOGFILE_SIZE | NUMBER | | Number of writes driven by the smallest redo log file size |
| WRITES_LOG_CHECKPOINT_SETTINGS | NUMBER | | Number of writes driven by the LOG_CHECKPOINT_INTERVAL parameter or the LOG_CHECKPOINT_TIMEOUT parameter |
| WRITES_OTHER_SETTINGS | NUMBER | | Number of writes driven by other reasons (such as the deprecated FAST_START_IO_TARGET parameter) |
| WRITES_AUTOTUNE | NUMBER | | Number of writes due to auto-tune checkpointing |
| WRITES_FULL_THREAD_CKPT | NUMBER | | Number of writes due to full thread checkpoints |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$INSTANCE_RECOVERY"

# DBA_HIST_INTERCONNECT_PINGS

DBA_HIST_INTERCONNECT_PINGS displays information about measured latency of interconnect messages (round-trip) from instance to instance. In Oracle Database 11*g* and later releases, the PING process assesses the latencies associated with communications for each pair of instances.

Every few seconds, the process in one instance (INSTANCE_NUMBER value) sends two messages to each instance (TARGET_INSTANCE value). One message has a size of 500 bytes and the other has a size of 8 KB. The message is received by the PING process on the target instance and is immediately acknowledged. The time for the round-trip is measured and collected.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| TARGET_INSTANCE | NUMBER | NOT NULL | Target instance number |
| CNT_500B | NUMBER | | Number of pings of size 500 bytes from INSTANCE_NUMBER to TARGET_INSTANCE since the startup of the source instance (INSTANCE_NUMBER) |
| WAIT_500B | NUMBER | | Sum of round-trip times for messages of size 500 bytes from INSTANCE_NUMBER to TARGET_INSTANCE since the startup of the source instance (INSTANCE_NUMBER). Dividing by CNT_500B gives the average latency. |
| WAITSQ_500B | NUMBER | | Sum of squares (divided by 1000) of round-trip times for messages of size 500 bytes from INSTANCE_NUMBER to TARGET_INSTANCE since the startup of the source instance (INSTANCE_NUMBER). When used with CNT_500B and WAIT_500B, the standard deviation of the latency can be calculated. |
| CNT_8K | NUMBER | | Number of pings of size 8 KB from INSTANCE_NUMBER to TARGET_INSTANCE since the startup of the source instance (INSTANCE_NUMBER) |
| WAIT_8K | NUMBER | | Sum of round-trip times for messages of size 8 KB from INSTANCE_NUMBER to TARGET_INSTANCE since the startup of the source instance (INSTANCE_NUMBER). Dividing by CNT_8K gives the average latency. |
| WAITSQ_8K | NUMBER | | Sum of squares (divided by 1000) of round-trip times for messages of size 8 KB from INSTANCE_NUMBER to TARGET_INSTANCE since the startup of the source instance (INSTANCE_NUMBER). When used with CNT_8K and WAIT_8K, the standard deviation of the latency can be calculated. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_IOSTAT_DETAIL

DBA_HIST_IOSTAT_DETAIL displays I/O statistics aggregated by combination of file type and function (component). This view contains snapshots of V$IOSTAT_FILE and V$IOSTAT_FUNCTION.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| FUNCTION_ID | NUMBER | NOT NULL | Function ID |
| FUNCTION_NAME | VARCHAR2(30) | NOT NULL | Function name |
| FILETYPE_ID | NUMBER | NOT NULL | Type of file (for example, log file, data file, and so on) |
| FILETYPE_NAME | VARCHAR2(30) | NOT NULL | Name of the file, in the case of a data file or temp file. For all other files, a corresponding string to be displayed (for example, ARCHIVELOG). |
| SMALL_READ_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes written |
| SMALL_READ_REQS | NUMBER | NOT NULL | Number of single block read requests |
| SMALL_WRITE_REQS | NUMBER | NOT NULL | Number of single block write requests |
| LARGE_READ_REQS | NUMBER | NOT NULL | Number of multiblock read requests |
| LARGE_WRITE_REQS | NUMBER | NOT NULL | Number of multiblock write requests |
| NUMBER_OF_WAITS | NUMBER | NOT NULL | Number of I/O waits by functionality |
| WAIT_TIME | NUMBER | NOT NULL | Total wait time (in milliseconds) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$IOSTAT_FILE"
- "V$IOSTAT_FUNCTION"

# DBA_HIST_IOSTAT_FILETYPE

DBA_HIST_IOSTAT_FILETYPE displays historical I/O statistics by file type. This view contains snapshots of V$IOSTAT_FILE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| FILETYPE_ID | NUMBER | NOT NULL | Type of file (for example, log file, data file, and so on) |
| FILETYPE_NAME | VARCHAR2(30) | NOT NULL | Name of the file, in the case of a data file or temp file. For all other files, a corresponding string to be displayed (for example, ARCHIVELOG). |
| SMALL_READ_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes written |
| SMALL_READ_REQS | NUMBER | NOT NULL | Number of single block read requests |
| SMALL_WRITE_REQS | NUMBER | NOT NULL | Number of single block write requests |
| SMALL_SYNC_READ_REQS | NUMBER | NOT NULL | Number of synchronous single block read requests |
| LARGE_READ_REQS | NUMBER | NOT NULL | Number of multiblock read requests |
| LARGE_WRITE_REQS | NUMBER | NOT NULL | Number of multiblock write requests |
| SMALL_READ_SERVICETIME | NUMBER | NOT NULL | Total service time (in milliseconds) for single block read requests |
| SMALL_WRITE_SERVICETIME | NUMBER | NOT NULL | Total service time (in milliseconds) for single block write requests |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SMALL_SYNC_READ_LATENCY | NUMBER | NOT NULL | Latency for single block synchronous reads (in milliseconds) |
| LARGE_READ_SERVICETIME | NUMBER | NOT NULL | Total service time (in milliseconds) for multiblock read requests |
| LARGE_WRITE_SERVICETIME | NUMBER | NOT NULL | Total service time (in milliseconds) for multiblock write requests |
| RETRIES_ON_ERROR | NUMBER | NOT NULL | Number of read retries on error |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$IOSTAT_FILE"

# DBA_HIST_IOSTAT_FILETYPE_NAME

DBA_HIST_IOSTAT_FILETYPE_NAME displays historical I/O statistics for file type names. This view contains snapshots of V$IOSTAT_FILE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| FILETYPE_ID | NUMBER | NOT NULL | Type of file (for example, log file, data file, and so on) |
| FILETYPE_NAME | VARCHAR2(30) | NOT NULL | Name of the file, in the case of a data file or temp file. For all other files, a corresponding string to be displayed (for example, ARCHIVELOG). |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$IOSTAT_FILE"

# DBA_HIST_IOSTAT_FUNCTION

DBA_HIST_IOSTAT_FUNCTION displays historical I/O statistics by function. This view contains snapshots of V$IOSTAT_FUNCTION.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| FUNCTION_ID | NUMBER | NOT NULL | Function ID |
| FUNCTION_NAME | VARCHAR2(30) | NOT NULL | Function name |
| SMALL_READ_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes written |
| SMALL_READ_REQS | NUMBER | NOT NULL | Number of single block read requests |
| SMALL_WRITE_REQS | NUMBER | NOT NULL | Number of single block write requests |
| LARGE_READ_REQS | NUMBER | NOT NULL | Number of multiblock read requests |
| LARGE_WRITE_REQS | NUMBER | NOT NULL | Number of multiblock write requests |
| NUMBER_OF_WAITS | NUMBER | NOT NULL | Number of I/O waits by functionality |
| WAIT_TIME | NUMBER | NOT NULL | Total wait time (in milliseconds) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$IOSTAT_FUNCTION"

# DBA_HIST_IOSTAT_FUNCTION_NAME

DBA_HIST_IOSTAT_FUNCTION_NAME displays historical I/O statistics by function names. This view contains snapshots of V$IOSTAT_FUNCTION.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| FUNCTION_ID | NUMBER | NOT NULL | Function ID |
| FUNCTION_NAME | VARCHAR2(30) | NOT NULL | Function name |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$IOSTAT_FUNCTION"

# DBA_HIST_JAVA_POOL_ADVICE

DBA_HIST_JAVA_POOL_ADVICE displays historical information about estimated parse time in the Java pool for different pool sizes. This view contains snapshots of V$JAVA_POOL_ADVICE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JAVA_POOL_SIZE_FOR_ESTIMATE | NUMBER | NOT NULL | Java pool size for the estimate (in megabytes) |
| JAVA_POOL_SIZE_FACTOR | NUMBER | | Size factor with respect to the current Java pool size |
| ESTD_LC_SIZE | NUMBER | | Estimated memory in use by the library cache (in megabytes) |
| ESTD_LC_MEMORY_OBJECTS | NUMBER | | Estimated number of library cache memory objects in the Java pool of the specified size |
| ESTD_LC_TIME_SAVED | NUMBER | | Estimated elapsed parse time saved (in seconds), owing to library cache memory objects being found in a Java pool of the specified size. This is the time that would have been spent in reloading the required objects in the Java pool had they been aged out due to insufficient amount of available free memory. |
| ESTD_LC_TIME_SAVED_FACTOR | NUMBER | | Estimated parse time saved factor with respect to the current Java pool size |
| ESTD_LC_LOAD_TIME | NUMBER | | Estimated elapsed time (in seconds) for parsing in a Java pool of the specified size. |
| ESTD_LC_LOAD_TIME_FACTOR | NUMBER | | Estimated load time factor with respect to the current Java pool size |
| ESTD_LC_MEMORY_OBJECT_HITS | NUMBER | | Estimated number of times a library cache memory object was found in a Java pool of the specified size |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | T.he ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$JAVA_POOL_ADVICE"

# 5

# Static Data Dictionary Views: DBA_HIST_LATCH to DBA_STORED_SETTINGS

This chapter contains the static data dictionary views DBA_HIST_LATCH through DBA_STORED_SETTINGS.

## DBA_HIST_LATCH

DBA_HIST_LATCH displays historical aggregate latch statistics for both parent and child latches, grouped by latch name. This view contains snapshots of V$LATCH.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| LATCH_HASH | NUMBER | NOT NULL | Latch hash |
| LATCH_NAME | VARCHAR2(64) | NOT NULL | Latch name |
| LEVEL# | NUMBER | | Latch level |
| GETS | NUMBER | | Number of times the latch was requested in willing-to-wait mode |
| MISSES | NUMBER | | Number of times the latch was requested in willing-to-wait mode and the requester had to wait |
| SLEEPS | NUMBER | | Number of times a willing-to-wait latch request resulted in a session sleeping while waiting for the latch |
| IMMEDIATE_GETS | NUMBER | | Number of times a latch was requested in no-wait mode |
| IMMEDIATE_MISSES | NUMBER | | Number of times a no-wait latch request did not succeed (that is, missed) |
| SPIN_GETS | NUMBER | | Number of willing-to-wait latch requests which missed the first try but succeeded while spinning |
| SLEEP[1 \| 2 \| 3 \| 4] | NUMBER | | These columns have been deprecated and are present only for compatibility with previous releases of Oracle. No data is accumulated for |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | these columns; they will always have a value of zero. |
| WAIT_TIME | NUMBER | | Elapsed time spent waiting for the latch (in microseconds) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LATCH"

# DBA_HIST_LATCH_CHILDREN

DBA_HIST_LATCH_CHILDREN displays historical statistics about child latches. This view includes all columns of DBA_HIST_LATCH plus the CHILD# column. Note that child latches have the same parent if their LATCH# columns match each other. This view contains snapshots of V$LATCH_CHILDREN.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| LATCH_HASH | NUMBER | NOT NULL | Latch hash |
| LATCH_NAME | VARCHAR2(64) | NOT NULL | Latch name |
| CHILD# | NUMBER | NOT NULL | Child latch number (unique only to each parent latch) |
| GETS | NUMBER | | Number of times the latch was requested in willing-to-wait mode |
| MISSES | NUMBER | | Number of times the latch was requested in willing-to-wait mode and the requester had to wait |
| SLEEPS | NUMBER | | Number of times a willing-to-wait latch request resulted in a session sleeping while waiting for the latch |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| IMMEDIATE_GETS | NUMBER | | Number of times a latch was requested in no-wait mode |
| IMMEDIATE_MISSES | NUMBER | | Number of times a no-wait latch request did not succeed (that is, missed) |
| SPIN_GETS | NUMBER | | Number of willing-to-wait latch requests which missed the first try but succeeded while spinning |
| SLEEP[1 \| 2 \| 3 \| 4] | NUMBER | | These columns have been deprecated and are present only for compatibility with previous releases of Oracle. No data is accumulated for these columns; they will always have a value of zero. |
| WAIT_TIME | NUMBER | | Elapsed time spent waiting for the latch (in microseconds) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "DBA_HIST_LATCH"
- "V$LATCH"

# DBA_HIST_LATCH_MISSES_SUMMARY

DBA_HIST_LATCH_MISSES_SUMMARY displays historical summary statistics about missed attempts to acquire a latch. This view contains snapshots of V$LATCH_MISSES.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| PARENT_NAME | VARCHAR2(50) | NOT NULL | Latch name of a parent latch |
| WHERE_IN_CODE | VARCHAR2(64) | NOT NULL | Location that attempted to acquire the latch |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NWFAIL_COUNT | NUMBER | | Number of times that no-wait acquisition of the latch failed |
| SLEEP_COUNT | NUMBER | | Number of times that acquisition attempts caused sleeps |
| WTR_SLP_COUNT | NUMBER | | Number of times a waiter slept |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LATCH_MISSES"

# DBA_HIST_LATCH_NAME

DBA_HIST_LATCH_NAME displays information about decoded latch names for the latches shown in DBA_HIST_LATCH. This view contains a snapshot of V$LATCHNAME.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| LATCH_HASH | NUMBER | NOT NULL | Latch hash |
| LATCH_NAME | VARCHAR2(64) | NOT NULL | Latch name |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "DBA_HIST_LATCH"
- "V$LATCHNAME"

# DBA_HIST_LATCH_PARENT

DBA_HIST_LATCH_PARENT displays historical statistics about parent latches. This view contains snapshots of V$LATCH_PARENT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| LATCH_HASH | NUMBER | NOT NULL | Latch hash |
| LATCH_NAME | VARCHAR2(64) | NOT NULL | Latch name |
| LEVEL# | NUMBER | NOT NULL | Latch level |
| GETS | NUMBER | | Number of times the latch was requested in willing-to-wait mode |
| MISSES | NUMBER | | Number of times the latch was requested in willing-to-wait mode and the requester had to wait |
| SLEEPS | NUMBER | | Number of times a willing-to-wait latch request resulted in a session sleeping while waiting for the latch |
| IMMEDIATE_GETS | NUMBER | | Number of times a latch was requested in no-wait mode |
| IMMEDIATE_MISSES | NUMBER | | Number of times a no-wait latch request did not succeed (that is, missed) |
| SPIN_GETS | NUMBER | | Number of willing-to-wait latch requests which missed the first try but succeeded while spinning |
| SLEEP[1 \| 2 \| 3 \| 4] | NUMBER | | These columns have been deprecated and are present only for compatibility with previous releases of Oracle. No data is accumulated for these columns; they will always have a value of zero. |
| WAIT_TIME | NUMBER | | Elapsed time spent waiting for the latch (in microseconds) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LATCH_PARENT"

# DBA_HIST_LIBRARYCACHE

DBA_HIST_LIBRARYCACHE displays historical statistics about library cache performance and activity. This view contains snapshots of V$LIBRARYCACHE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| NAMESPACE | VARCHAR2(15) | NOT NULL | Library cache namespace |
| GETS | NUMBER | | Number of times a lock was requested for objects of the namespace |
| GETHITS | NUMBER | | Number of times an object's handle was found in memory |
| PINS | NUMBER | | Number of times a PIN was requested for objects of the namespace |
| PINHITS | NUMBER | | Number of times all of the metadata pieces of the library object were found in memory |
| RELOADS | NUMBER | | Any PIN of an object that is not the first PIN performed since the object handle was created, and which requires loading the object from disk |
| INVALIDATIONS | NUMBER | | Total number of times objects in the namespace were marked invalid because a dependent object was modified |
| DLM_LOCK_REQUESTS | NUMBER | | Number of GET requests lock instance locks |
| DLM_PIN_REQUESTS | NUMBER | | Number of PIN requests lock instance locks |
| DLM_PIN_RELEASES | NUMBER | | Number of release requests PIN instance locks |
| DLM_INVALIDATION_REQUESTS | NUMBER | | Number of GET requests for invalidation instance locks |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DLM_INVALIDATIONS | NUMBER | | Number of invalidation pings received from other instances |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LIBRARYCACHE"

# DBA_HIST_LOG

DBA_HIST_LOG displays historical log file information from the control file. This view contains snapshots of V$LOG.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| GROUP# | NUMBER | NOT NULL | Log group number |
| THREAD# | NUMBER | NOT NULL | Log thread number |
| SEQUENCE# | NUMBER | NOT NULL | Log sequence number |
| BYTES | NUMBER | | Size of the log (in bytes) |
| MEMBERS | NUMBER | | Number of members in the log group |
| ARCHIVED | VARCHAR2(3) | | Archive status (YES) or NO) |
| STATUS | VARCHAR2(16) | | Log status:<br>• UNUSED - Online redo log has never been written to. This is the state of a redo log that was just added, or just after a RESETLOGS, when it is not the current redo log.<br>• CURRENT - Current redo log. This implies that the redo log is active. The redo log could be open or closed.<br>• ACTIVE - Log is active but is not the current log. It is needed for crash recovery. It may be |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | in use for block recovery. It may or may not be archived.<br>• CLEARING - Log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, the status changes to UNUSED.<br>• CLEARING_CURRENT - Current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch such as an I/O error writing the new log header.<br>• INACTIVE - Log is no longer needed for instance recovery. It may be in use for media recovery. It may or may not be archived.<br>• INVALIDATED - Archived the current redo log without a log switch. |
| FIRST_CHANGE# | NUMBER | | Lowest system change number (SCN) in the log |
| FIRST_TIME | DATE | | Time of the first SCN in the log |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LOG"

# DBA_HIST_MEM_DYNAMIC_COMP

DBA_HIST_MEM_DYNAMIC_COMP displays historical memory component sizes. This view contains snapshots of V$MEMORY_DYNAMIC_COMPONENTS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| COMPONENT | VARCHAR2(64) | NOT NULL | Component name |
| CURRENT_SIZE | NUMBER | | Current size of the component |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MIN_SIZE | NUMBER | | Minimum size of the component since instance startup |
| MAX_SIZE | NUMBER | | Maximum size of the component since instance startup |
| USER_SPECIFIED_SIZE | NUMBER | | Value of the user parameter for the component |
| OPER_COUNT | NUMBER | | Number of operations since instance startup |
| LAST_OPER_TYPE | VARCHAR2(13) | | Last completed operation for the component:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW<br>• SHRINK<br>• SHRINK_CANCEL |
| LAST_OPER_MODE | VARCHAR2(9) | | Mode of the last completed operation:<br>• MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| LAST_OPER_TIME | DATE | | Start time of the last completed operation |
| GRANULE_SIZE | NUMBER | | Granularity of the GROW or SHRINK operation |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$MEMORY_DYNAMIC_COMPONENTS"

# DBA_HIST_MEMORY_RESIZE_OPS

DBA_HIST_MEMORY_RESIZE_OPS displays memory resize operations history. This view contains snapshots of V$MEMORY_RESIZE_OPS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| COMPONENT | VARCHAR2(64) | NOT NULL | Component name |
| OPER_TYPE | VARCHAR2(13) | NOT NULL | Operation type:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW<br>• SHRINK<br>• SHRINK_CANCEL |
| START_TIME | DATE | NOT NULL | Start time of the operation |
| END_TIME | DATE | NOT NULL | End time of the operation |
| TARGET_SIZE | NUMBER | NOT NULL | Requested value of the parameter after the resize |
| OPER_MODE | VARCHAR2(9) | | Operation mode:<br>• MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| PARAMETER | VARCHAR2(80) | | Name of the parameter for the resize operation |
| INITIAL_SIZE | NUMBER | | Parameter value at the start of the operation |
| FINAL_SIZE | NUMBER | | Real value of the parameter after the resize |
| STATUS | VARCHAR2(9) | | Completion status of the operation:<br>• INACTIVE<br>• PENDING<br>• COMPLETE<br>• CANCELLED<br>• ERROR |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$MEMORY_RESIZE_OPS"

# DBA_HIST_MEMORY_TARGET_ADVICE

DBA_HIST_MEMORY_TARGET_ADVICE displays memory target advice history. This view contains snapshots of V$MEMORY_TARGET_ADVICE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| MEMORY_SIZE | NUMBER | NOT NULL | If the MEMORY_SIZE_FACTOR column has a value of 1, then this column shows the current size of memory, as set by the MEMORY_TARGET initialization parameter.<br>If the value of the MEMORY_SIZE_FACTOR column is less than or greater than 1, then this column shows a proposed memory size. |
| MEMORY_SIZE_FACTOR | NUMBER | | A multiplier for the current memory size. Possible values are 0.25, 0.5, 0.75, 1, 1.5, 1.75 and 2. This multiplier times the current memory size equals the value of the MEMORY_SIZE column. |
| ESTD_DB_TIME | NUMBER | | For current memory size (MEMORY_SIZE_FACTOR = 1), the amount of database time required to complete the current workload. For a proposed memory size, the estimated amount of database time that would be required if the MEMORY_TARGET parameter were changed to the proposed size. |
| ESTD_DB_TIME_FACTOR | NUMBER | | For a proposed memory size, ratio of estimated database time to current database time |
| VERSION | NUMBER | | Version number of this recommendation |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$MEMORY_TARGET_ADVICE"

# DBA_HIST_METRIC_NAME

DBA_HIST_METRIC_NAME describes attributes of the set of RDBMS metrics. This view contains a snapshot of V$METRICNAME.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| GROUP_ID | NUMBER | NOT NULL | Metric Group ID |
| GROUP_NAME | VARCHAR2(64) | | Metric group name |
| METRIC_ID | NUMBER | NOT NULL | Metric ID |
| METRIC_NAME | VARCHAR2(64) | NOT NULL | Metric name |
| METRIC_UNIT | VARCHAR2(64) | NOT NULL | Unit of measurement |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$METRICNAME"

# DBA_HIST_MTTR_TARGET_ADVICE

DBA_HIST_MTTR_TARGET_ADVICE displays historical predictions of the number of physical I/O requests for the MTTR corresponding to each row. The data also includes a physical I/O factor, which is the ratio of the number of estimated I/O requests to the number of I/O requests actually performed by the current MTTR setting during the measurement interval. This view contains snapshots of V$MTTR_TARGET_ADVICE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| MTTR_TARGET_FOR_ESTIMATE | NUMBER | | MTTR setting being simulated (equal to the current MTTR setting if this is the first row of the view) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADVICE_STATUS | VARCHAR2(5) | | Current status of MTTR simulation:<br>• ON SET<br>• READY SET<br>• OFF |
| DIRTY_LIMIT | NUMBER | | Dirty buffer limit derived from the MTTR being simulated |
| ESTD_CACHE_WRITES | NUMBER | | Estimated number of cache physical writes under the MTTR |
| ESTD_CACHE_WRITE_FACTOR | NUMBER | | Estimated cache physical write ratio under the MTTR. It is the ratio of the estimated number of cache writes to the number of cache writes under the current MTTR setting. |
| ESTD_TOTAL_WRITES | NUMBER | | Estimated total number of physical writes under the MTTR |
| ESTD_TOTAL_WRITE_FACTOR | NUMBER | | Estimated total physical write ratio under the MTTR. It is the ratio of the estimated total number of physical writes to the total number of physical writes under the current MTTR setting. |
| ESTD_TOTAL_IOS | NUMBER | | Estimated total number of I/O requests under the MTTR |
| ESTD_TOTAL_IO_FACTOR | NUMBER | | Estimated total I/O ratio under the MTTR. It is the ratio of the estimated total number of I/O requests to the total number of I/O requests under the current MTTR setting. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$MTTR_TARGET_ADVICE"

# DBA_HIST_MUTEX_SLEEP

DBA_HIST_MUTEX_SLEEP displays mutex sleep summary historical statistics information.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database identifier for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| MUTEX_TYPE | VARCHAR2(32) | NOT NULL | Mutex type |
| LOCATION | VARCHAR2(40) | NOT NULL | The code location where the waiter slept for the mutex |
| SLEEPS | NUMBER | | Number of sleeps for this MUTEX_TYPE and LOCATION |
| WAIT_TIME | NUMBER | | Wait time in microseconds |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_OPTIMIZER_ENV

DBA_HIST_OPTIMIZER_ENV displays the optimizer environments that have been captured in the Workload Repository. This view is used with the DBA_HIST_SQLSTAT view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| OPTIMIZER_ENV_HASH_VALUE | NUMBER | NOT NULL | Hash value for the optimizer environment |
| OPTIMIZER_ENV | RAW(2000) | | Optimizer environment |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"DBA_HIST_SQLSTAT"

# DBA_HIST_OSSTAT

DBA_HIST_OSSTAT displays historical operating system statistics. This view contains snapshots of V$OSSTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| STAT_ID | NUMBER | NOT NULL | Statistic ID |
| STAT_NAME | VARCHAR2(64) | NOT NULL | Statistic name |
| VALUE | NUMBER | | Statistic value |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$OSSTAT"

# DBA_HIST_OSSTAT_NAME

DBA_HIST_OSSTAT_NAME displays the names of the operating system statistics. This view is used with DBA_HIST_OSSTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| STAT_ID | NUMBER | NOT NULL | Statistic ID |
| STAT_NAME | VARCHAR2(64) | NOT NULL | Statistic name |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"DBA_HIST_OSSTAT"

# DBA_HIST_PARAMETER

DBA_HIST_PARAMETER displays historical information about the initialization parameters that were in effect for the instance. This view contains snapshots of V$SYSTEM_PARAMETER.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| PARAMETER_HASH | NUMBER | NOT NULL | Parameter hash |
| PARAMETER_NAME | VARCHAR2(64) | NOT NULL | Name of the parameter |
| VALUE | VARCHAR2(512) | | Parameter value for the session (if modified within the session); otherwise, the instance-wide parameter value |
| ISDEFAULT | VARCHAR2(9) | | Indicates whether the parameter is set to the default value (TRUE) or the parameter value was specified in the parameter file (FALSE) |
| ISMODIFIED | VARCHAR2(10) | | Indicates whether the parameter has been modified after instance startup:<br>• MODIFIED - Parameter has been modified with ALTER SESSION<br>• SYSTEM_MOD - Parameter has been modified with ALTER SYSTEM (which causes all the currently logged in sessions' values to be modified)<br>• FALSE - Parameter has not been modified after instance startup |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYSTEM_PARAMETER"

# DBA_HIST_PARAMETER_NAME

DBA_HIST_PARAMETER_NAME displays information about the parameter names captured in the workload repository. This view is used with the DBA_HIST_PARAMETER view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| PARAMETER_HASH | NUMBER | NOT NULL | Parameter hash |
| PARAMETER_NAME | VARCHAR2(64) | NOT NULL | Name of the parameter |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"DBA_HIST_PARAMETER"

# DBA_HIST_PDB_INSTANCE

DBA_HIST_PDB_INSTANCE displays the pluggable databases (PDB)s and instances in the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STARTUP_TIME | TIMESTAMP(3) | NOT NULL | Startup time of the instance |
| CON_DBID | NUMBER | NOT NULL | The database ID of the PDB for the sampled session |
| OPEN_TIME | TIMESTAMP(3) | NOT NULL | Time the PDB was last opened |
| OPEN_MODE | VARCHAR2(16) | | Open mode of the database |
| PDB_NAME | VARCHAR2(128) | | PDB name |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_PERSISTENT_QMN_CACHE

DBA_HIST_PERSISTENT_QMN_CACHE displays the historical summary background queue table activity. This view contains snapshots from V$PERSISTENT_QMN_CACHE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| QUEUE_TABLE_ID | NUMBER | NOT NULL | Queue table object ID |
| TYPE | VARCHAR2(32) | | Type of the queue table's queue monitor cache |
| STATUS | NUMBER | | Status of the queue table's queue monitor cache |
| NEXT_SERVICE_TIME | TIMESTAMP(3) | | Time when the queue table should be serviced by QMON servers |
| WINDOW_END_TIME | TIMESTAMP(3) | | Time manager activity period for non-owner queue table operations |
| TOTAL_RUNS | NUMBER | | Total number of times this queue table is served |
| TOTAL_LATENCY | NUMBER | | Cumulative latency in serving the queue table (in hundredths of a second) |
| TOTAL_ELAPSED_TIME | NUMBER | | Total time spent in processing this queue table (in seconds) |
| TOTAL_CPU_TIME | NUMBER | | Cumulative CPU time for serving the queue table (in hundredths of a second) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TMGR_ROWS_PROCESSED | NUMBER | | Number of time manager entries processed |
| TMGR_ELAPSED_TIME | NUMBER | | Cumulative time for time management activities (in hundredths of a second) |
| TMGR_CPU_TIME | NUMBER | | Cumulative CPU time for time management activities (in hundredths of a second) |
| LAST_TMGR_PROCESSING_TIME | TIMESTAMP(3) | | Last timer manager processing time |
| DEQLOG_ROWS_PROCESSED | NUMBER | | Number of dequeue log entries processed |
| DEQLOG_PROCESSING_ELAPSED_TIME | NUMBER | | Total time for processing dequeue log entries (in hundredths of a second) |
| DEQLOG_PROCESSING_CPU_TIME | NUMBER | | Total CPU time for processing dequeue log entries (in hundredths of a second) |
| LAST_DEQLOG_PROCESSING_TIME | TIMESTAMP(3) | | Last dequeue log processing time |
| DEQUEUE_INDEX_BLOCKS_FREED | NUMBER | | Number of dequeue index blocks freed |
| HISTORY_INDEX_BLOCKS_FREED | NUMBER | | Number of history index blocks freed |
| TIME_INDEX_BLOCKS_FREED | NUMBER | | Number of time manager index blocks freed |
| INDEX_CLEANUP_COUNT | NUMBER | | Number of times index block cleanup was attempted |
| INDEX_CLEANUP_ELAPSED_TIME | NUMBER | | Total time for index block cleanup (in hundredths of a second) |
| INDEX_CLEANUP_CPU_TIME | NUMBER | | Total CPU time for index block cleanup (in hundredths of a second) |
| LAST_INDEX_CLEANUP_TIME | TIMESTAMP(3) | | Last index block cleanup time |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that `CON_DBID` identifies. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$PERSISTENT_QMN_CACHE"

# DBA_HIST_PERSISTENT_QUEUES

DBA_HIST_PERSISTENT_QUEUES displays STREAMS AQ persistent queues historical statistics information. This view contains snapshots of V$PERSISTENT_QUEUES.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| QUEUE_SCHEMA | VARCHAR2(128) | NOT NULL | Owner of the queue |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| QUEUE_ID | NUMBER | NOT NULL | Identifier for the queue |
| FIRST_ACTIVITY_TIME | TIMESTAMP(6) | | First queue activity time since database startup |
| ENQUEUED_MSGS | NUMBER | | Number of messages enqueued |
| DEQUEUED_MSGS | NUMBER | | Number of messages dequeued<br>**Note:** This column will not be incremented until all the subscribers of the message have dequeued the message and its retention time has elapsed. |
| BROWSED_MSGS | NUMBER | | Number of messages that have been browsed |
| ELAPSED_ENQUEUE_TIME | NUMBER | | Total time (in hundredths of a second) spent doing enqueue |
| ELAPSED_DEQUEUE_TIME | NUMBER | | Total time (in hundredths of a second) spent doing dequeue |
| ENQUEUE_CPU_TIME | NUMBER | | Total CPU time for enqueue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | | Total CPU time for dequeue (in hundredths of a second) |
| AVG_MSG_AGE | NUMBER | | Average age of messages in the queue |
| DEQUEUED_MSG_LATENCY | NUMBER | | Last dequeued message latency (in seconds) |
| ELAPSED_TRANSFORMATION_TIME | NUMBER | | Total time (in hundredths of a second) spent doing transformation |
| ELAPSED_RULE_EVALUATION_TIME | NUMBER | | Total time (in hundredths of a second) spent doing rule evaluation |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ENQUEUED_EXPIRY_MSGS | NUMBER | | Number of messages enqueued with expiry |
| ENQUEUED_DELAY_MSGS | NUMBER | | Number of messages enqueued with delay |
| MSGS_MADE_EXPIRED | NUMBER | | Number of messages expired by time manager |
| MSGS_MADE_READY | NUMBER | | Number of messages made ready by time manager |
| LAST_ENQUEUE_TIME | TIMESTAMP(6) | | Last message enqueue time |
| LAST_DEQUEUE_TIME | TIMESTAMP(6) | | Last message dequeue time |
| LAST_TM_EXPIRY_TIME | TIMESTAMP(6) | | Last time message was expired by time manager |
| LAST_TM_READY_TIME | TIMESTAMP(6) | | Last time message was made ready by time manager |
| ENQUEUE_TRANSACTIONS | NUMBER | | Number of enqueue transactions |
| DEQUEUE_TRANSACTIONS | NUMBER | | Number of dequeue transactions |
| EXECUTION_COUNT | NUMBER | | Number of executions of the dequeue cursor |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$PERSISTENT_QUEUES"

# DBA_HIST_PERSISTENT_SUBS

DBA_HIST_PERSISTENT_SUBS displays STREAMS AQ persistent queue subscribers historical statistics information. This view contains snapshots of V$PERSISTENT_SUBSCRIBERS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| QUEUE_SCHEMA | VARCHAR2(128) | NOT NULL | Owner of the queue |
| QUEUE_NAME | VARCHAR2(128) | NOT NULL | Name of the queue |
| SUBSCRIBER_ID | NUMBER | NOT NULL | Internal subscriber number |
| SUBSCRIBER_NAME | VARCHAR2(128) | | Name of the subscriber |
| SUBSCRIBER_ADDRESS | VARCHAR2(1024) | | Address of the subscribing agent |
| SUBSCRIBER_TYPE | VARCHAR2(128) | | Type of the subscriber:<br>• PROXY - Propagation subscriber<br>• SUBSCRIBER - Normal subscriber<br>• RECIPIENT - Recipient |
| FIRST_ACTIVITY_TIME | TIMESTAMP(6) | | First subscriber activity time since database startup |
| ENQUEUED_MSGS | NUMBER | | Number of messages enqueued since FIRST_ACTIVITY_TIME |
| DEQUEUED_MSGS | NUMBER | | Number of messages dequeued since FIRST_ACTIVITY_TIME |
| AVG_MSG_AGE | NUMBER | | Average age of messages in the queue |
| BROWSED_MSGS | NUMBER | | Number of messages that have been browsed |
| EXPIRED_MSGS | NUMBER | | Number of messages expired since FIRST_ACTIVITY_TIME |
| DEQUEUED_MSG_LATENCY | NUMBER | | Last dequeued message latency (in seconds) |
| LAST_ENQUEUE_TIME | TIMESTAMP(6) | | Timestamp of the last enqueued message |
| LAST_DEQUEUE_TIME | TIMESTAMP(6) | | Timestamp of the last dequeued message |
| ELAPSED_DEQUEUE_TIME | NUMBER | | Total time spent in dequeue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | | Total CPU time for dequeue (in hundredths of a second) |
| DEQUEUE_TRANSACTIONS | NUMBER | | Number of dequeue transactions |
| EXECUTION_COUNT | NUMBER | | Number of executions of the dequeue index cursor |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$PERSISTENT_SUBSCRIBERS"

# DBA_HIST_PGA_TARGET_ADVICE

DBA_HIST_PGA_TARGET_ADVICE displays historical predictions of how the cache hit percentage and over allocation count statistics displayed by the V$PGASTAT performance view would be impacted if the value of the PGA_AGGREGATE_TARGET parameter is changed. This view contains snapshots of V$PGA_TARGET_ADVICE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| PGA_TARGET_FOR_ESTIMATE | NUMBER | NOT NULL | Value of PGA_AGGREGATE_TARGET for the prediction (in bytes) |
| PGA_TARGET_FACTOR | NUMBER | | PGA_TARGET_FOR_ESTIMATE / the current value of the PGA_AGGREGATE_TARGET parameter |
| ADVICE_STATUS | VARCHAR2(3) | | Indicates whether the advice is enabled (ON) or disabled (OFF) depending on the value of the STATISTICS_LEVEL parameter |
| BYTES_PROCESSED | NUMBER | | Total bytes processed by all the work areas considered by this advice (in bytes) |
| ESTD_TIME | NUMBER | | Time (in seconds) required to process the bytes |
| ESTD_EXTRA_BYTES_RW | NUMBER | | Estimated number of extra bytes which would be read or written if PGA_AGGREGATE_TARGET was set to the value of the PGA_TARGET_FOR_ESTIMATE column. This number is derived from the estimated number and size of work areas which would run in one-pass (or multi-pass) for that value of PGA_AGGREGATE_TARGET. |
| ESTD_PGA_CACHE_HIT_PERCENTAGE | NUMBER | | Estimated value of the cache hit percentage statistic when PGA_AGGREGATE_TARGET equals PGA_TARGET_FOR_ESTIMATE. This column is derived from the above two columns and is equal |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | to BYTES_PROCESSED / (BYTES_PROCESSED + ESTD_EXTRA_BYTES_RW) |
| ESTD_OVERALLOC_COUNT | NUMBER | | Estimated number of PGA memory over-allocations if the value of PGA_AGGREGATE_TARGET is set to PGA_TARGET_FOR_ESTIMATE. A nonzero value means that PGA_TARGET_FOR_ESTIMATE is not large enough to run the work area workload. Hence, PGA_AGGREGATE_TARGET should not be set to PGA_TARGET_FOR_ESTIMATE since Oracle will not be able to honor that target. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$PGASTAT"
- "V$PGA_TARGET_ADVICE"
- "PGA_AGGREGATE_TARGET"

# DBA_HIST_PGASTAT

DBA_HIST_PGASTAT displays historical PGA memory usage statistics as well as statistics about the automatic PGA memory manager when it is enabled. This view contains snapshots of V$PGASTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Unique snapshot ID |
| NAME | VARCHAR2(64) | NOT NULL | Name of the statistic:<br>• aggregate PGA auto target<br>• aggregate PGA target parameter<br>• bytes processed<br>• cache hit percentage<br>• extra bytes read/written<br>• global memory bound |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • max processes count<br>• maximum PGA allocated<br>• maximum PGA used for auto workareas<br>• maximum PGA used for manual workareas<br>• over allocation count<br>• PGA memory freed back to OS<br>• process count<br>• recompute count (total)<br>• total freeable PGA memory<br>• total PGA allocated<br>• total PGA inuse<br>• total PGA used for auto workareas<br>• total PGA used for manual workareas<br>**See Also:** V$PGASTAT for descriptions of the statistics |
| VALUE | NUMBER | | Statistic value |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$PGASTAT"

# DBA_HIST_PLAN_OPERATION_NAME

DBA_HIST_PLAN_OPERATION_NAME displays historical information about SQL plan operation names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database identifier |
| OPERATION_ID | NUMBER | NOT NULL | Plan operation identifier |
| OPERATION_NAME | VARCHAR2(64) | | Plan operation name. This value also appears in the SQL_PLAN_OPERATION column of the DBA_HIST_ACTIVE_SESS_HISTORY view. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_PLAN_OPTION_NAME

DBA_HIST_PLAN_OPTION_NAME displays historical information about SQL plan option names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database identifier |
| OPTION_ID | NUMBER | NOT NULL | Plan option identifier |
| OPTION_NAME | VARCHAR2(64) | | Plan option name. This value also appears in the SQL_PLAN_OPTIONS column of the DBA_HIST_ACTIVE_SESS_HISTORY view. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_PROCESS_MEM_SUMMARY

DBA_HIST_PROCESS_MEM_SUMMARY displays historical information about dynamic PGA memory usage by named component categories for each process.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| CATEGORY | VARCHAR2(15) | NOT NULL | Category name. Categories include "SQL", "PL/SQL", "OLAP" and "JAVA". Special categories are "Freeable" and "Other". Freeable memory has |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | been allocated to the process by the operating system, but has not been allocated to a category. "Other" memory has been allocated to a category, but not to one of the named categories |
| NUM_PROCESSES | NUMBER | | Number of processes |
| NON_ZERO_ALLOCS | NUMBER | | Number of processes with nonzero allocations |
| USED_TOTAL | NUMBER | | Bytes of PGA memory used by the process for the category |
| ALLOCATED_TOTAL | NUMBER | | Total number of bytes of PGA memory allocated by the process for the category. |
| ALLOCATED_AVG | NUMBER | | Average number of bytes of PGA memory allocated by the process for the category |
| ALLOCATED_STDDEV | NUMBER | | Standard deviation of the number of bytes of PGA memory allocated by the process for the category |
| ALLOCATED_MAX | NUMBER | | Maximum bytes of PGA memory ever allocated by the process for the category |
| MAX_ALLOCATED_MAX | NUMBER | | Maximum bytes of PGA memory that can be allocated by the process for the category |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_REPLICATION_TBL_STATS

DBA_HIST_REPLICATION_TBL_STATS displays replication table statistics for Oracle GoldenGate and XStream sessions. This view is intended for use with Automatic Workload Repository (AWR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| APPLY_NAME | VARCHAR2(128) | NOT NULL | Name of the apply process |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_OWNER | VARCHAR2(128) | NOT NULL | Owner of the table |
| SESSION_MODULE | VARCHAR2(64) | NOT NULL | Session module. Valid values:<br>• XStream<br>• GoldenGate |
| TOTAL_INSERTS | NUMBER | | Number of insert operations on this table processed by this apply server |
| TOTAL_UPDATES | NUMBER | | Number of update operations on this table processed by this apply server |
| TOTAL_DELETES | NUMBER | | Number of delete operations on this table processed by this apply server |
| CDR_SUCCESSFUL | NUMBER | | Number of successfully resolved conflicts |
| CDR_FAILED | NUMBER | | Number of conflicts that could not be resolved due to an error during resolution |
| REPERR_CNT | NUMBER | | The total number of errors for the replication operation |
| HANDLE_COLLISIONS | NUMBER | | Number of collisions on this table handled by this apply server |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_REPLICATION_TXN_STATS

DBA_HIST_REPLICATION_TXN_STATS displays replication transaction statistics for Oracle GoldenGate and XStream sessions.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| OBJECT_NAME | VARCHAR2(30) | NOT NULL | Object name |
| SESSION_TYPE | VARCHAR2(64) | NOT NULL | Type of session |
| SESSION_MODULE | VARCHAR2(64) | NOT NULL | Session module. Valid values:<br>• XStream |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • GoldenGate |
| SOURCE_DATABASE | VARCHAR2(128) | | Database where the transaction originated |
| SOURCE_TXN_ID | VARCHAR2(128) | NOT NULL | Original transaction ID at the source database |
| FIRST_LCR_TIME | DATE | | Time of the first LCR (message in an error transaction) |
| TOTAL_LCRS_COUNT | NUMBER | | Total number of LCRs for this replication transaction |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_REPORTS

DBA_HIST_REPORTS displays information about XML reports captured into Automatic Workload Repository (AWR). The reports themselves belong to components such as SQL Monitor, DBOP, and Real-Time ADDM.

Each XML report contains details about some activity of a component. For example, a SQL Monitor report contains a detailed report about a particular execution of a SQL statement, or a Real-Time ADDM report contains system performance data analyzed by Real-Time ADDM.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | | ID of the first Automatic Workload Repository (AWR) snapshot that will be taken after this report is generated |
| DBID | NUMBER | | Database ID of the current database for the report |
| INSTANCE_NUMBER | NUMBER | | Instance number (for an Oracle RAC system) |
| REPORT_ID | NUMBER | | ID of the captured report |
| COMPONENT_ID | NUMBER | | ID of the component (for example, SQL Monitor) whose report is captured |
| SESSION_ID | NUMBER | | ID of the session corresponding to the captured report (currently used only for SQL Monitor reports) |
| SESSION_SERIAL# | NUMBER | | Session serial number corresponding to the captured report (currently used only for SQL Monitor reports) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PERIOD_START_TIME | DATE | | Time when the activity period started |
| PERIOD_END_TIME | DATE | | Time when the activity period ended |
| GENERATION_TIME | DATE | | Time when this report was generated |
| COMPONENT_NAME | VARCHAR2(128) | | Name of the component whose report this is |
| REPORT_NAME | VARCHAR2(128) | | Name of this report |
| REPORT_PARAMETERS | VARCHAR2(1024) | | Parameters associated with this report |
| KEY1 | VARCHAR2(128) | | Key1 associated with the captured report |
| KEY2 | VARCHAR2(128) | | Key2 associated with the captured report |
| KEY3 | VARCHAR2(128) | | Key3 associated with the captured report |
| KEY4 | VARCHAR2(256) | | Key4 associated with the captured report |
| GENERATION_COST_SECONDS | NUMBER | | Time taken to generate this report (in seconds) |
| REPORT_SUMMARY | VARCHAR2(4000) | | Summary of this report |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID[1] | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DBA_HIST_REPORTS_DETAILS"

# DBA_HIST_REPORTS_CONTROL

DBA_HIST_REPORTS_CONTROL contains control information about the report capture mechanism that automatically captures XML reports to Automatic Workload Repository (AWR). Reports are captured automatically for components like SQL Monitor and Real-Time Automatic Database Diagnostic Monitor (Real-Time ADDM).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | | Database ID of the current database for the report |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXECUTION_MODE | VARCHAR2(12) | | Mode of execution of automatic report capture. Possible values:<br>• REGULAR: Regular per-minute report capture subject to DBTIME budget<br>• FULL_CAPTURE: Report capture will be run per minute without the DBTIME budget constraints and is provided to capture a more comprehensive set of reports<br>**NOTE**: The FULL_CAPTURE mode can be started and ended respectively by executing the START_REPORT_CAPTURE and FINISH_REPORT_CAPTURE APIs in the DBMS_AUTO_REPORT package. At all other times, the execution mode should be REGULAR. |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_AUTO_REPORT package

# DBA_HIST_REPORTS_DETAILS

DBA_HIST_REPORTS_DETAILS displays details about each report captured in Automatic Workload Repository (AWR). Metadata for each report appears in the DBA_HIST_REPORTS view while the actual report is available in the DBA_HIST_REPORTS_DETAILS view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | | ID of the first AWR snapshot that will be taken after this report is generated |
| DBID | NUMBER | | Database ID of the current database for the report |
| INSTANCE_NUMBER | NUMBER | | Instance number (for an Oracle RAC system) |
| REPORT_ID | NUMBER | | ID of the captured report |
| SESSION_ID | NUMBER | | ID of the session corresponding to the captured report (currently used only for SQL Monitor reports) |
| SESSION_SERIAL# | NUMBER | | Session serial number relevant to this report (currently used only for SQL Monitor reports) |
| GENERATION_TIME | DATE | | Time when this report was generated |
| REPORT_COMPRESSED | BLOB | | Actual XML report in compressed form |
| REPORT | CLOB | | Full uncompressed report |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID[1] | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DBA_HIST_REPORTS"

# DBA_HIST_REPORTS_TIMEBANDS

DBA_HIST_REPORTS_TIMEBANDS contains bands of time with a new row created every day corresponding to a band of time. Each band of time represents a period of time, and has a row for every report captured automatically into Automatic Workload Repository (AWR) during that time. If the activity period of a report spans across two bands of time (for example. the activity started before midnight and ended after midnight), then the view contains two rows for that report, with one row for each band of time. The view is partitioned to provide fast access to all reports captured in a given time frame.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | | The AWR snapshot id corresponding to the report |
| DBID | NUMBER | | Database ID of the current database for the report |
| INSTANCE_NUMBER | NUMBER | | Instance number (for an Oracle RAC system) |
| CON_DBID | NUMBER | | CDB ID of the captured report |
| COMPONENT_ID | NUMBER | | ID of the component (for example, SQL Monitor) whose report is captured |
| COMPONENT_NAME | VARCHAR2(128) | | Name of the component whose report is captured |
| BAND_START_TIME | DATE | | Starting time of the time band |
| BAND_LENGTH | NUMBER | | Length of time band in days (currently unused) |
| REPORT_ID | NUMBER | | ID of the captured report |
| REPORT_GENERATION_TIME | DATE | | Time when the report was generated |
| PERIOD_START_TIME | DATE | | Time when the activity period started |
| PERIOD_END_TIME | DATE | | Time when the activity period ended |
| KEY1 | VARCHAR2(128) | | Key1 associated with the captured report |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| KEY2 | VARCHAR2(128) | | Key2 associated with the captured report |
| KEY3 | VARCHAR2(128) | | Key3 associated with the captured report |
| KEY4 | VARCHAR2(256) | | Key4 associated with the captured report |
| SESSION_ID | NUMBER | | ID of the session corresponding to the captured report (currently used only for SQL Monitor reports) |
| SESSION_SERIAL# | NUMBER | | Session serial number corresponding to the captured report (currently used only for SQL Monitor reports) |
| CON_ID[1] | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# DBA_HIST_RESOURCE_LIMIT

DBA_HIST_RESOURCE_LIMIT displays historical information about global resource use for some of the system resource. This view contains snapshots of V$RESOURCE_LIMIT.

If time is of interest, join this view with DBA_HIST_SNAPSHOT.END_INTERVAL_TIME.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| RESOURCE_NAME | VARCHAR2(30) | NOT NULL | Name of the resource |
| CURRENT_UTILIZATION | NUMBER | | Number of (resources, locks, or processes) currently being used |
| MAX_UTILIZATION | NUMBER | | Maximum consumption of the resource since the last instance start up |
| INITIAL_ALLOCATION | VARCHAR2(10) | | Initial allocation. This will be equal to the value specified for the resource in the initialization parameter file (UNLIMITED for infinite allocation). |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LIMIT_VALUE | VARCHAR2(10) | | Unlimited for resources and locks. This can be greater than the initial allocation value (UNLIMITED for infinite limit). |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_ROWCACHE_SUMMARY

DBA_HIST_ROWCACHE_SUMMARY displays historical summary statistics for data dictionary activity. This view contains snapshots of V$ROWCACHE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| PARAMETER | VARCHAR2(32) | NOT NULL | Name of the initialization parameter that determines the number of entries in the data dictionary cache |
| TOTAL_USAGE | NUMBER | | Sum of the total number of entries in the cache |
| USAGE | NUMBER | | Number of cache entries that contain valid data |
| GETS | NUMBER | | Total number of requests for information on the data object |
| GETMISSES | NUMBER | | Number of data requests resulting in cache misses |
| SCANS | NUMBER | | Number of scan requests |
| SCANMISSES | NUMBER | | Number of times a scan failed to find the data in the cache |
| SCANCOMPLETES | NUMBER | | For a list of subordinate entries, the number of times the list was scanned completely |
| MODIFICATIONS | NUMBER | | Number of inserts, updates, and deletions |
| FLUSHES | NUMBER | | Number of times flushed to disk |
| DLM_REQUESTS | NUMBER | | Number of DLM requests |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DLM_CONFLICTS | NUMBER | | Number of DLM conflicts |
| DLM_RELEASES | NUMBER | | Number of DLM releases |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$ROWCACHE"

# DBA_HIST_RSRC_CONSUMER_GROUP

DBA_HIST_RSRC_CONSUMER_GROUP displays historical information about Resource Manager consumer groups. This view contains snapshots of V$RSRC_CONS_GROUP_HISTORY.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SEQUENCE# | NUMBER | NOT NULL | A sequential counter that uniquely describes the DBA_HIST_RSRC_PLAN entry to which these consumer group statistics apply. When the instance is restarted, this value is reset to zero. |
| CONSUMER_GROUP_ID | NUMBER | NOT NULL | Consumer group object ID (a unique number, consistent across database shutdowns and startups) |
| CONSUMER_GROUP_NAME | VARCHAR2(128) | NOT NULL | Name of the consumer group |
| REQUESTS | NUMBER | NOT NULL | Cumulative number of requests that were executed in the consumer group |
| CPU_WAIT_TIME | NUMBER | NOT NULL | Cumulative amount of time that sessions waited for CPU because of resource management. This does not include waits due to latch or enqueue contention, I/O waits, and so on. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CPU_WAITS | NUMBER | NOT NULL | Cumulative number of times all sessions in the consumer group had to wait for CPU because of resource management. This does not include waits due to latch or enqueue contention, I/O waits, and so on. |
| CONSUMED_CPU_TIME | NUMBER | NOT NULL | Cumulative amount of CPU time consumed by all sessions in the consumer group (in milliseconds) |
| YIELDS | NUMBER | NOT NULL | Cumulative number of times that sessions in the consumer group had to yield CPU to other sessions because of quantum expiration |
| ACTIVE_SESS_LIMIT_HIT | NUMBER | NOT NULL | Number of times that sessions in the consumer group were queued because the consumer group reached its active session limit |
| UNDO_LIMIT_HIT | NUMBER | NOT NULL | Number of times that queries in the consumer group were cancelled because the consumer group reached its UNDO_POOL limit |
| SWITCHES_IN_CPU_TIME | NUMBER | NOT NULL | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_TIME limit |
| SWITCHES_OUT_CPU_TIME | NUMBER | NOT NULL | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_TIME limit |
| SWITCHES_IN_IO_MEGABYTES | NUMBER | NOT NULL | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_MEGABYTES limit |
| SWITCHES_OUT_IO_MEGABYTES | NUMBER | NOT NULL | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_MEGABYTES limit |
| SWITCHES_IN_IO_REQUESTS | NUMBER | NOT NULL | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_REQS limit |
| SWITCHES_OUT_IO_REQUESTS | NUMBER | NOT NULL | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_REQS limit |
| SQL_CANCELED | NUMBER | NOT NULL | Number of times that SQL queries running in the consumer group were aborted because they exceeded the Resource Manager plan's SWITCH_TIME limit and CANCEL_SQL was specified as the Resource Manager plan's SWITCH_GROUP |
| ACTIVE_SESS_KILLED | NUMBER | NOT NULL | Number of times that sessions running in the consumer group were terminated because they exceeded the Resource Manager plan's SWITCH_TIME limit and KILL_SESSION was specified as the Resource Manager plan's SWITCH_GROUP |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| IDLE_SESS_KILLED | NUMBER | NOT NULL | Number of times that sessions in the consumer group were killed because they were idle for too long (reached MAX_IDLE_TIME) |
| IDLE_BLKR_SESS_KILLED | NUMBER | NOT NULL | Number of times that sessions in the consumer group were killed because they were idle too long (reached MAX_IDLE_BLOCKER_TIME) and were blocking other sessions |
| QUEUED_TIME | NUMBER | NOT NULL | Total amount of time that sessions in the consumer group have spent in the QUEUED state because of the active session limit (in milliseconds) |
| QUEUE_TIME_OUTS | NUMBER | NOT NULL | Number of times that requests from sessions in the consumer group timed out because they were queued for too long (reached QUEUEING_P1) |
| IO_SERVICE_TIME | NUMBER | NOT NULL | Cumulative I/O wait time (in milliseconds) |
| IO_SERVICE_WAITS | NUMBER | NOT NULL | Total number of wait requests |
| SMALL_READ_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | NOT NULL | Number of multiblock megabytes written |
| SMALL_READ_REQUESTS | NUMBER | NOT NULL | Number of single block read requests |
| SMALL_WRITE_REQUESTS | NUMBER | NOT NULL | Number of single block write requests |
| LARGE_READ_REQUESTS | NUMBER | NOT NULL | Number of multiblock read requests |
| LARGE_WRITE_REQUESTS | NUMBER | NOT NULL | Number of multiblock write requests |
| PQS_QUEUED | NUMBER | | Number of times that sessions in the consumer group were queued when trying to run parallel statements |
| PQ_QUEUED_TIME | NUMBER | | Total amount of time that sessions in the consumer group were queued when trying to run parallel statements (in milliseconds) |
| PQ_QUEUE_TIME_OUTS | NUMBER | | Number of times that parallel statements from sessions in the consumer group timed out because their queue time exceeded the Resource Manager plan's PARALLEL_QUEUE_TIMEOUT limit |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PQS_COMPLETED | NUMBER | | Total number of completed parallel statements in the consumer group |
| PQ_SERVERS_USED | NUMBER | | Total number of parallel servers used by completed parallel statements in the consumer group |
| PQ_ACTIVE_TIME | NUMBER | | Cumulative sum of the parallel active times for all completed parallel statements in the consumer group (in milliseconds) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$RSRC_CONS_GROUP_HISTORY"

# DBA_HIST_RSRC_PLAN

DBA_HIST_RSRC_PLAN displays historical information about resource plans. This view contains snapshots of V$RSRC_PLAN_HISTORY.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SEQUENCE# | NUMBER | NOT NULL | A sequential counter that uniquely describes a row. When the instance is restarted, this value is reset to zero. |
| START_TIME | DATE | NOT NULL | Time that the resource plan was enabled |
| END_TIME | DATE | | Time that the resource plan was disabled; NULL if the row contains the current resource plan information |
| PLAN_ID | NUMBER | NOT NULL | Resource plan ID; NULL if the Resource Manager was disabled |
| PLAN_NAME | VARCHAR2(128) | NOT NULL | Resource plan name; NULL if the Resource Manager was disabled |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CPU_MANAGED | VARCHAR2(4) | NOT NULL | Indicates whether the resource plan has parameters that specify a policy for how the Resource Manager should schedule sessions to manage CPU usage (ON) or whether Resource Manager is not managing CPU usage (OFF) |
| PARALLEL_EXECUTION_MANAGED | VARCHAR2(4) | | State of parallel statement queuing:<br>• OFF - Parallel statement queuing is disabled<br>• STARTUP - Parallel statement queuing is enabled. This is a temporary state that can occur when an Oracle RAC database is undergoing configuration changes<br>• FIFO - Parallel statement queuing is enabled. All parallel statements are managed in a single Oracle RAC FIFO queue<br>• FULL - Parallel statement queuing is enabled. All parallel statements are managed in per-consumer group queues according to the current resource plan. This state is used when a resource plan that contains resource allocation directives (MGMT_P*) is enabled. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$RSRC_PLAN_HISTORY"

# DBA_HIST_RULE_SET

DBA_HIST_RULE_SET displays historical information about rule set statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the rule set |
| NAME | VARCHAR2(128) | NOT NULL | Name of the rule set |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STARTUP_TIME | DATE | NOT NULL | Startup time of the instance |
| CPU_TIME | NUMBER | | Total CPU time (in hundredths of a second) spent in evaluation of the rule set |
| ELAPSED_TIME | NUMBER | | Total elapsed time (in hundredths of a second) spent in evaluation of the rule set |
| EVALUATIONS | NUMBER | | Number of evaluations on the rule set |
| SQL_FREE_EVALUATIONS | NUMBER | | Number of evaluations on the rule set which did not internally issue SQL to evaluate rules |
| SQL_EXECUTIONS | NUMBER | | Total number of SQL statements executed during evaluation of the rule set |
| RELOADS | NUMBER | | Number of times the rule set object was reloaded in shared memory |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SEG_STAT

DBA_HIST_SEG_STAT displays historical information about segment-level statistics. This view captures the top segments based on a set of criteria and captures information from V$SEGSTAT. The total value is the value of the statistics since instance startup. The delta value is the value of the statistics from the BEGIN_INTERVAL_TIME to the END_INTERVAL_TIME in the DBA_HIST_SNAPSHOT view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| TS# | NUMBER | NOT NULL | Tablespace number |
| OBJ# | NUMBER | NOT NULL | Dictionary object number |
| DATAOBJ# | NUMBER | NOT NULL | Data object number |
| LOGICAL_READS_TOTAL | NUMBER | | Cumulative value for logical reads |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOGICAL_READS_DELTA | NUMBER | | Delta value for logical reads |
| BUFFER_BUSY_WAITS_TOTAL | NUMBER | | Cumulative value for buffer busy waits |
| BUFFER_BUSY_WAITS_DELTA | NUMBER | | Delta value for buffer busy waits |
| DB_BLOCK_CHANGES_TOTAL | NUMBER | | Cumulative value for db block changes |
| DB_BLOCK_CHANGES_DELTA | NUMBER | | Delta value for db block changes |
| PHYSICAL_READS_TOTAL | NUMBER | | Cumulative value for physical reads |
| PHYSICAL_READS_DELTA | NUMBER | | Delta value for physical reads |
| PHYSICAL_WRITES_TOTAL | NUMBER | | Cumulative value for physical writes |
| PHYSICAL_WRITES_DELTA | NUMBER | | Delta value for physical writes |
| PHYSICAL_READS_DIRECT_TOTAL | NUMBER | | Cumulative value for physical reads direct |
| PHYSICAL_READS_DIRECT_DELTA | NUMBER | | Delta value for physical reads direct |
| PHYSICAL_WRITES_DIRECT_TOTAL | NUMBER | | Cumulative value for physical writes direct |
| PHYSICAL_WRITES_DIRECT_DELTA | NUMBER | | Delta value for physical writes direct |
| ITL_WAITS_TOTAL | NUMBER | | Cumulative value for ITL waits |
| ITL_WAITS_DELTA | NUMBER | | Delta value for ITL waits |
| ROW_LOCK_WAITS_TOTAL | NUMBER | | Cumulative value for row lock waits |
| ROW_LOCK_WAITS_DELTA | NUMBER | | Delta value for row lock waits |
| GC_CR_BLOCKS_SERVED_TOTAL | NUMBER | | Cumulative value for global cache CR blocks served |
| GC_CR_BLOCKS_SERVED_DELTA | NUMBER | | Delta value for global cache CR blocks served |
| GC_CU_BLOCKS_SERVED_TOTAL | NUMBER | | Cumulative value for global cache current blocks served |
| GC_CU_BLOCKS_SERVED_DELTA | NUMBER | | Delta value for global cache current blocks served |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GC_BUFFER_BUSY_TOTAL | NUMBER | | Cumulative value for global cache buffer busy |
| GC_BUFFER_BUSY_DELTA | NUMBER | | Delta value for global cache buffer busy |
| GC_CR_BLOCKS_RECEIVED_TOTAL | NUMBER | | Cumulative value for global cache CR blocks received |
| GC_CR_BLOCKS_RECEIVED_DELTA | NUMBER | | Delta value for global cache CR blocks received |
| GC_CU_BLOCKS_RECEIVED_TOTAL | NUMBER | | Cumulative value for global cache current blocks received |
| GC_CU_BLOCKS_RECEIVED_DELTA | NUMBER | | Delta value for global cache current blocks received |
| SPACE_USED_TOTAL | NUMBER | | Cumulative value for space used |
| SPACE_USED_DELTA | NUMBER | | Delta value for space used |
| SPACE_ALLOCATED_TOTAL | NUMBER | | Cumulative value for space allocated |
| SPACE_ALLOCATED_DELTA | NUMBER | | Delta value for space allocated |
| TABLE_SCANS_TOTAL | NUMBER | | Cumulative value for table scans |
| TABLE_SCANS_DELTA | NUMBER | | Delta value for table scans |
| CHAIN_ROW_EXCESS_TOTAL | NUMBER | | Cumulative value of number of chained row pieces that can be eliminated by table reorganization |
| CHAIN_ROW_EXCESS_DELTA | NUMBER | | Delta value of number of chained row pieces that can be eliminated by table reorganization |
| PHYSICAL_READ_REQUESTS_TOTAL | NUMBER | | Cumulative value of number of physical read I/O requests issued for the monitored segment |
| PHYSICAL_READ_REQUESTS_DELTA | NUMBER | | Delta value of number of physical read I/O requests issued for the monitored segment |
| PHYSICAL_WRITE_REQUESTS_TOTAL | NUMBER | | Cumulative value of number of physical write I/O requests issued for the monitored segment |
| PHYSICAL_WRITE_REQUESTS_DELTA | NUMBER | | Delta value of number of physical write I/O requests issued for the monitored segment |
| OPTIMIZED_PHYSICAL_READS_TOTAL | NUMBER | | Cumulative value of number of physical reads from Database Smart Flash Cache for the monitored segment |
| OPTIMIZED_PHYSICAL_READS_DELTA | NUMBER | | Delta value of number of physical reads from Database Smart Flash Cache for the monitored segment |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SEGSTAT"
- "DBA_HIST_SNAPSHOT"

# DBA_HIST_SEG_STAT_OBJ

DBA_HIST_SEG_STAT_OBJ displays all the names of the segments captured in the workload repository. This view is used with the DBA_HIST_SEG_STAT view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| TS# | NUMBER | NOT NULL | Tablespace number |
| OBJ# | NUMBER | NOT NULL | Dictionary object number |
| DATAOBJ# | NUMBER | NOT NULL | Data object number |
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject (for example: partition) |
| OBJECT_TYPE | VARCHAR2(18) | | Type of the object for example: table, tablespace) |
| TABLESPACE_NAME | VARCHAR2(30) | | Tablespace Name for the object |
| PARTITION_TYPE | VARCHAR2(8) | | Partition Type, if relevant |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"DBA_HIST_SEG_STAT"

# DBA_HIST_SERVICE_NAME

DBA_HIST_SERVICE_NAME displays the names of the Services tracked by the Workload Repository. This view contains information for V$SERVICES.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| SERVICE_NAME_HASH | NUMBER | NOT NULL | Hash of the service name |
| SERVICE_NAME | VARCHAR2(64) | NOT NULL | Name of the service |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SERVICES"

# DBA_HIST_SERVICE_STAT

DBA_HIST_SERVICE_STAT displays the history of important service statistics tracked by the Workload Repository. The call rate statistics in this view can be used for making run-time routing decisions, for tracking service levels, and for per-instance diagnostics per call rate.

The elapsed timing for each call provides a relative value across instances for how well a node is processing SQL calls issued under a service name. When aggregation is enabled for the service name, this view provides historical data on the timing and work done for calls issued for the whole service. This view contains information from V$SERVICE_STATS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SERVICE_NAME_HASH | NUMBER | NOT NULL | Hash of the service name |
| SERVICE_NAME | VARCHAR2(64) | NOT NULL | Name of the service |
| STAT_ID | NUMBER | NOT NULL | Statistic identifier |
| STAT_NAME | VARCHAR2(64) | NOT NULL | Statistic name |
| VALUE | NUMBER | | Value of the statistic |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SERVICE_STATS"

# DBA_HIST_SERVICE_WAIT_CLASS

DBA_HIST_SERVICE_WAIT_CLASS displays the history of wait class information for services as tracked by the Workload Repository. This view contains information from V$SERVICE_WAIT_CLASS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SERVICE_NAME_HASH | NUMBER | NOT NULL | Hash of the service name |
| SERVICE_NAME | VARCHAR2(64) | NOT NULL | Name of the service |
| WAIT_CLASS_ID | NUMBER | NOT NULL | Identifier for the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | | Name for the class of the wait event |
| TOTAL_WAITS | NUMBER | | Total number of waits for this event |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TIME_WAITED | NUMBER | | Total amount of time waited for this event (in hundredths of a second) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SERVICE_WAIT_CLASS"

# DBA_HIST_SESS_SGA_STATS

DBA_HIST_SESS_SGA_STATS shows usage statistics for high utilization Oracle GoldenGate and XStream sessions.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Object name |
| SESSION_TYPE | VARCHAR2(64) | NOT NULL | Type of session |
| SESSION_MODULE | VARCHAR2(64) | NOT NULL | Session module. Valid values:<br>• XStream<br>• GoldenGate |
| SGA_USED | NUMBER | | The total amount of shared memory (in bytes) currently used by the session out of the amount allocated (SGA_ALLOCATED) |
| SGA_ALLOCATED | NUMBER | | The total amount of shared memory (in bytes) allocated from the pool for the session |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SESS_TIME_STATS

DBA_HIST_SESS_TIME_STATS displays information about CPU and I/O time for interesting Streams sessions.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SESSION_TYPE | VARCHAR2(64) | NOT NULL | Type of session |
| MIN_LOGON_TIME | DATE | | Minimum logon time |
| SUM_CPU_TIME | NUMBER | | Total CPU time |
| SUM_SYS_IO_WAIT | NUMBER | | Total system I/O wait time |
| SUM_USER_IO_WAIT | NUMBER | | Total user I/O wait time |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| SESSION_MODULE | VARCHAR2(64) | NOT NULL | Session module. Valid values:<br>• Streams<br>• XStream<br>• GoldenGate |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SESSMETRIC_HISTORY

DBA_HIST_SESSMETRIC_HISTORY displays the history of several important session metrics.

**Note:**

This view is populated only if a session metric exceeds a server metric threshold that was configured using the DBMS_SERVER_ALERT package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| BEGIN_TIME | DATE | NOT NULL | Begin time of the interval |
| END_TIME | DATE | NOT NULL | End time of the interval |
| SESSID | NUMBER | NOT NULL | Session ID |
| SERIAL# | NUMBER | NOT NULL | Session serial number |
| INTSIZE | NUMBER | NOT NULL | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | NOT NULL | Group ID |
| METRIC_ID | NUMBER | NOT NULL | Metric ID |
| METRIC_NAME | VARCHAR2(64) | NOT NULL | Metric name |
| VALUE | NUMBER | NOT NULL | Metric Value |
| METRIC_UNIT | VARCHAR2(64) | NOT NULL | Unit of measurement |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

The DBMS_SERVER_ALERT package in *Oracle Database PL/SQL Packages and Types Reference*

# DBA_HIST_SGA

DBA_HIST_SGA displays historical summary information about the system global area (SGA). This view contains snapshots of V$SGA.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(64) | NOT NULL | SGA component group |
| VALUE | NUMBER | NOT NULL | Memory size (in bytes) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SGA"

# DBA_HIST_SGA_TARGET_ADVICE

DBA_HIST_SGA_TARGET_ADVICE provides historical information about the SGA_TARGET initialization parameter.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SGA_SIZE | NUMBER | NOT NULL | Size of the SGA |
| SGA_SIZE_FACTOR | NUMBER | NOT NULL | Ratio between the SGA_SIZE and the current size of the SGA |
| ESTD_DB_TIME | NUMBER | NOT NULL | Estimated DB_TIME for this SGA_SIZE |
| ESTD_PHYSICAL_READS | NUMBER | | Estimated number of physical reads |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"SGA_TARGET"

# DBA_HIST_SGASTAT

DBA_HIST_SGASTAT displays detailed historical information on the system global area (SGA). This view contains snapshots of V$SGASTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| NAME | VARCHAR2(64) | | SGA component group |
| POOL | VARCHAR2(12) | | Designates the pool in which the memory in NAME resides:<br>• shared pool - Memory is allocated from the shared pool<br>• large pool - Memory is allocated from the large pool<br>• java pool - Memory is allocated from the Java pool<br>• streams pool - Memory is allocated from the Streams pool |
| BYTES | NUMBER | | Memory size (in bytes) |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SGASTAT"

# DBA_HIST_SHARED_POOL_ADVICE

DBA_HIST_SHARED_POOL_ADVICE displays historical information about estimated parse time in the shared pool for different pool sizes. This view contains snapshots of V$SHARED_POOL_ADVICE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SHARED_POOL_SIZE_FOR_ESTIMATE | NUMBER | NOT NULL | Shared pool size for the estimate (in megabytes) |
| SHARED_POOL_SIZE_FACTOR | NUMBER | | Size factor with respect to the current shared pool size |
| ESTD_LC_SIZE | NUMBER | | Estimated memory in use by the library cache (in megabytes) |
| ESTD_LC_MEMORY_OBJECTS | NUMBER | | Estimated number of library cache memory objects in the shared pool of the specified size |
| ESTD_LC_TIME_SAVED | NUMBER | | Estimated elapsed parse time saved (in seconds), owing to library cache memory objects being found in a shared pool of the specified size. This is the time that would have been spent in reloading the required objects in the shared pool had they been aged out due to insufficient amount of available free memory. |
| ESTD_LC_TIME_SAVED_FACTOR | NUMBER | | Estimated parse time saved factor with respect to the current shared pool size |
| ESTD_LC_LOAD_TIME | NUMBER | | Estimated elapsed time (in seconds) for parsing in a shared pool of the specified size. |
| ESTD_LC_LOAD_TIME_FACTOR | NUMBER | | Estimated load time factor with respect to the current shared pool size |
| ESTD_LC_MEMORY_OBJECT_HITS | NUMBER | | Estimated number of times a library cache memory object was found in a shared pool of the specified size |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that `CON_DBID` identifies. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SHARED_POOL_ADVICE"

# DBA_HIST_SHARED_SERVER_SUMMARY

DBA_HIST_SHARED_SERVER_SUMMARY displays historical information for shared servers. This includes information about shared server activity, the servers, common queues, and dispatcher queues. This view obtains information from V$SHARED_SERVER, V$DISPATCHER, V$CIRCUIT, and V$QUEUE, and is aggregated over all servers, dispatchers, queues, and circuits.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| NUM_SAMPLES | NUMBER | | Total number of samples |
| SAMPLE_TIME | NUMBER | | Last sample timestamp |
| SAMPLED_TOTAL_CONN | NUMBER | | Cumulative sum of total number of connections over all samples. To determine the average number of connections between two snapshots, divide the difference in SAMPLED_TOTAL_CONN by the difference in NUM_SAMPLES. |
| SAMPLED_ACTIVE_CONN | NUMBER | | Cumulative sum of active number of connections over all samples. To determine the average number of active connections between two snapshots, divide the difference in SAMPLED_ACTIVE_CONN by the difference in NUM_SAMPLES. |
| SAMPLED_TOTAL_SRV | NUMBER | | Cumulative sum of total number of servers over all samples. To determine the average number of servers between two snapshots, divide the difference in SAMPLED_TOTAL_SRV by the difference in NUM_SAMPLES. |
| SAMPLED_ACTIVE_SRV | NUMBER | | Cumulative sum of active number of servers over all samples. To determine the average number of active servers between two snapshots, divide the difference in SAMPLED_ACTIVE_SRV by the difference in NUM_SAMPLES. |
| SAMPLED_TOTAL_DISP | NUMBER | | Cumulative sum of total number of dispatchers over all samples. To determine the average number of dispatchers between two snapshots, divide the difference in SAMPLED_TOTAL_DISP by the difference in NUM_SAMPLES. |
| SAMPLED_ACTIVE_DISP | NUMBER | | Cumulative sum of active number of dispatchers over all samples. To determine the average number of active dispatchers between two snapshots, divide the difference in SAMPLED_ACTIVE_DISP by the difference in NUM_SAMPLES. |
| SRV_BUSY | NUMBER | | Total shared server busy time (in hundredths of a second) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SRV_IDLE | NUMBER | | Total shared server idle time (in hundredths of a second) |
| SRV_IN_NET | NUMBER | | Total shared server incoming network wait time (in hundredths of a second). This includes waits for receives and resets. This time is also included in SRV_BUSY. |
| SRV_OUT_NET | NUMBER | | Total shared server outgoing network wait time (in hundredths of a second). This includes waits for sends and outbound connection requests. This time is also included in SRV_BUSY. |
| SRV_MESSAGES | NUMBER | | Number of messages processed |
| SRV_BYTES | NUMBER | | Total number of bytes in all messages |
| CQ_WAIT | NUMBER | | Total time that all items in the common queue have waited (in hundredths of a second) |
| CQ_TOTALQ | NUMBER | | Total number of items that have ever been in the common queue |
| DQ_TOTALQ | NUMBER | | Total number of items that have ever been in a dispatcher queue |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SHARED_SERVER"
- "V$DISPATCHER"
- "V$CIRCUIT"
- "V$QUEUE"

# DBA_HIST_SNAP_ERROR

DBA_HIST_SNAP_ERROR displays information about the snapshot error information in the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table in which the error occurred |
| ERROR_NUMBER | NUMBER | NOT NULL | Error number for the error encountered |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SNAPSHOT

DBA_HIST_SNAPSHOT displays information about the snapshots in the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| STARTUP_TIME | TIMESTAMP(3) | NOT NULL | Startup time of the instance |
| BEGIN_INTERVAL_TIME | TIMESTAMP(3) | NOT NULL | Time at the beginning of the snapshot interval |
| END_INTERVAL_TIME | TIMESTAMP(3) | NOT NULL | Time at the end of the snapshot interval; the actual time the snapshot was taken |
| FLUSH_ELAPSED | INTERVAL DAY(5) TO SECOND(1) | | Amount of time to perform the snapshot |
| SNAP_LEVEL | NUMBER | | Snapshot level |
| ERROR_COUNT | NUMBER | | Number of errors occurring in the tables for the particular snapshot |
| SNAP_FLAG | NUMBER | | Condition under which the snapshot was inserted. Possible values are:<br>0 - Snapshot was taken automatically by the Manageability Monitor Process (MMON process)<br>1 - Manual snapshot created using a PL/SQL package<br>2 - Imported snapshot |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | 4 - Snapshot taken while Diagnostic Pack or Tuning Pack was not enabled |
| SNAP_TIMEZONE | INTERVAL DAY(0) TO SECOND(0) | | Snapshot time zone expressed as offset from UTC (Coordinated Universal Time) time zone |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

Table F-1 for more information about the MMON process

# DBA_HIST_SQL_BIND_METADATA

DBA_HIST_SQL_BIND_METADATA displays historical information on metadata for bind variables used by SQL cursors.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| NAME | VARCHAR2(128) | | Name of the bind variable |
| POSITION | NUMBER | NOT NULL | Position of the bind variable in the SQL statement |
| DUP_POSITION | NUMBER | | If the binding is performed by name and the bind variable is duplicated, then this column gives the position of the primary bind variable |
| DATATYPE | NUMBER | | Internal identifier for the bind data type. Beginning in Oracle Database 12*c*, a number representing a PL/SQL data type can appear in this column. |
| DATATYPE_STRING | VARCHAR2(15) | | Textual representation of the bind data type. Beginning in Oracle Database 12*c*, a text representation of a PL/SQL-only data type can appear in this column. If the actual data type is a PL/SQL sub type, the name of the data type, not the sub type will be displayed. |
| CHARACTER_SID | NUMBER | | National character set identifier |
| PRECISION | NUMBER | | Precision (for numeric binds) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCALE | NUMBER | | Scale (for numeric binds) |
| MAX_LENGTH | NUMBER | | Maximum bind length |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SQL_PLAN

DBA_HIST_SQL_PLAN displays the execution plan information for each child cursor in the workload repository. This view captures information from V$SQL_PLAN and is used with the DBA_HIST_SQLSTAT view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Numerical representation of the SQL plan for the cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). |
| ID | NUMBER | NOT NULL | A number assigned to each step in the execution plan |
| OPERATION | VARCHAR2(30) | | Name of the internal operation performed in this step (for example, TABLE ACCESS) |
| OPTIONS | VARCHAR2(30) | | A variation on the operation described in the OPERATION column (for example, FULL) |
| OBJECT_NODE | VARCHAR2(128) | | Name of the database link used to reference the object (a table name or view name). For local queries that use parallel execution, this column describes the order in which output from operations is consumed. |
| OBJECT# | NUMBER | | Object number of the table or the index |
| OBJECT_OWNER | VARCHAR2(128) | | Name of the user who owns the schema containing the table or index |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or index |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_ALIAS | VARCHAR2(261) | | Alias for the object |
| OBJECT_TYPE | VARCHAR2(20) | | Type of the object |
| OPTIMIZER | VARCHAR2(20) | | Current mode of the optimizer for the first row in the plan (statement line), for example, CHOOSE. When the operation is a database access (for example, TABLE ACCESS), this column indicates whether or not the object is analyzed. |
| PARENT_ID | NUMBER | | ID of the next execution step that operates on the output of the current step |
| DEPTH | NUMBER | | Depth (or level) of the operation in the tree. It is not necessary to issue a CONNECT BY statement to get the level information, which is generally used to indent the rows from the PLAN_TABLE table. The root operation (statement) is level 0. |
| POSITION | NUMBER | | Order of processing for all operations that have the same PARENT_ID |
| SEARCH_COLUMNS | NUMBER | | Number of index columns with start and stop keys (that is, the number of columns with matching predicates) |
| COST | NUMBER | | Cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| CARDINALITY | NUMBER | | Estimate, by the cost-based optimizer, of the number of rows produced by the operation |
| BYTES | NUMBER | | Estimate, by the cost-based optimizer, of the number of bytes produced by the operation |
| OTHER_TAG | VARCHAR2(35) | | Describes the contents of the OTHER column. See EXPLAIN PLAN for values. |
| PARTITION_START | VARCHAR2(64) | | Start partition of a range of accessed partitions |
| PARTITION_STOP | VARCHAR2(64) | | Stop partition of a range of accessed partitions |
| PARTITION_ID | NUMBER | | Step that computes the pair of values of the PARTITION_START and PARTITION_STOP columns |
| OTHER | VARCHAR2(4000) | | Other information specific to the execution step that users may find useful. See EXPLAIN PLAN for values. |
| DISTRIBUTION | VARCHAR2(20) | | Stores the method used to distribute rows from producer query servers to consumer query servers |
| CPU_COST | NUMBER | | CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| IO_COST | NUMBER | | I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| TEMP_SPACE | NUMBER | | Temporary space usage of the operation (sort or hash-join) as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| ACCESS_PREDICATES | VARCHAR2(4000) | | Predicates used to locate rows in an access structure. For example, start or stop predicates for an index range scan. |
| FILTER_PREDICATES | VARCHAR2(4000) | | Predicates used to filter rows before producing them |
| PROJECTION | VARCHAR2(4000) | | Expressions produced by the operation |
| TIME | NUMBER | | Elapsed time (in seconds) of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| QBLOCK_NAME | VARCHAR2(128) | | Name of the query block |
| REMARKS | VARCHAR2(4000) | | Remarks |
| TIMESTAMP | DATE | | Timestamp for when the plan was produced |
| OTHER_XML | CLOB | | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML because it allows multiple pieces of information to be stored, including the following:<br>• Name of the schema against which the query was parsed<br>• Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan<br>• Indication of whether or not dynamic statistics were used to produce the plan<br>• The outline data, a set of optimizer hints that can be used to regenerate the same plan |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SQL_PLAN"
- "DBA_HIST_SQLSTAT"

# DBA_HIST_SQL_SUMMARY

DBA_HIST_SQL_SUMMARY displays historical SQL summary information.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| TOTAL_SQL | NUMBER | NOT NULL | Total number of SQLs |
| TOTAL_SQL_MEM | NUMBER | NOT NULL | Total sharable memory in bytes for SQLs |
| SINGLE_USE_SQL | NUMBER | NOT NULL | Total number of single execution SQLs |
| SINGLE_USE_SQL_MEM | NUMBER | NOT NULL | Total sharable memory in bytes for single execution SQLs |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SQL_WORKAREA_HSTGRM

DBA_HIST_SQL_WORKAREA_HSTGRM displays the historical cumulative work area execution statistics (cumulated since instance startup) for different work area groups. This view contains snapshots of V$SQL_WORKAREA_HISTOGRAM.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| LOW_OPTIMAL_SIZE | NUMBER | NOT NULL | Lower bound for the optimal memory requirement of work areas included in the row (in bytes) |
| HIGH_OPTIMAL_SIZE | NUMBER | NOT NULL | Upper bound for the optimal memory requirement of work areas included in the row (in bytes) |
| OPTIMAL_EXECUTIONS | NUMBER | | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which have been executed in optimal mode since instance startup |
| ONEPASS_EXECUTIONS | NUMBER | | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which have been executed in one-pass mode since instance startup |
| MULTIPASSES_EXECUTIONS | NUMBER | | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which have been executed in multi-pass mode since instance startup |
| TOTAL_EXECUTIONS | NUMBER | | Sum of OPTIMAL_EXECUTIONS, ONEPASS_EXECUTIONS, and MULTIPASSES_EXECUTIONS |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SQL_WORKAREA_HISTOGRAM"

# DBA_HIST_SQLBIND

DBA_HIST_SQLBIND displays historical information on bind variables used by SQL cursors. This view contains snapshots of V$SQL_BIND_CAPTURE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| NAME | VARCHAR2(128) | | Name of the bind variable |
| POSITION | NUMBER | NOT NULL | Position of the bind variable in the SQL statement |
| DUP_POSITION | NUMBER | | If the binding is performed by name and the bind variable is duplicated, then this column gives the position of the primary bind variable. |
| DATATYPE | NUMBER | | Internal identifier for the bind data type. Beginning in Oracle Database 12*c*, a number representing a PL/SQL data type can appear in this column. |
| DATATYPE_STRING | VARCHAR2(15) | | Textual representation of the bind data type. Beginning in Oracle Database 12*c*, a text representation of a PL/SQL-only data type can appear in this column. If the actual data type is a PL/SQL sub type, the name of the data type, not the sub type will be displayed. |
| CHARACTER_SID | NUMBER | | National character set identifier |
| PRECISION | NUMBER | | Precision (for numeric binds) |
| SCALE | NUMBER | | Scale (for numeric binds) |
| MAX_LENGTH | NUMBER | | Maximum bind length |
| WAS_CAPTURED | VARCHAR2(3) | | Indicates whether the bind value was captured (YES) or not (NO) |
| LAST_CAPTURED | DATE | | Date when the bind value was captured. Bind values are captured when SQL statements are executed. To limit the overhead, binds are captured at most every 15 minutes for a given cursor. |
| VALUE_STRING | VARCHAR2(4000) | | Value of the bind represented as a string |
| VALUE_ANYDATA | ANYDATA | | Value of the bind represented using the self-descriptive Sys.AnyData data type. This representation is useful to programmatically decode the value of the bind variable. This column is NULL if a PL/SQL-only data type appears in the DATATYPE column. |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SQL_BIND_CAPTURE"

# DBA_HIST_SQLCOMMAND_NAME

DBA_HIST_SQLCOMMAND_NAME displays the mapping between SQL opcodes and names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| COMMAND_TYPE | NUMBER | NOT NULL | SQL command number |
| COMMAND_NAME | VARCHAR2(64) | | SQL command name |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SQLSTAT

DBA_HIST_SQLSTAT displays historical information about SQL statistics. This view captures the top SQL statements based on a set of criteria and captures the statistics information from V$SQL. The total value is the value of the statistics since instance startup. The delta value is the value of the statistics from the BEGIN_INTERVAL_TIME to the END_INTERVAL_TIME in the DBA_HIST_SNAPSHOT view.

This view is used with the DBA_HIST_OPTIMIZER_ENV, DBA_HIST_SQLTEXT, and DBA_HIST_SQL_PLAN views to provide a complete picture of historical SQL statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Numerical representation of the SQL plan for the cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). |
| OPTIMIZER_COST | NUMBER | | Cost of the query given by the optimizer |
| OPTIMIZER_MODE | VARCHAR2(10) | | Mode under which the SQL statement is executed |
| OPTIMIZER_ENV_HASH_VALUE | NUMBER | | Hash Value for the optimizer environment |
| SHARABLE_MEM | NUMBER | | Amount of shared memory used by the child cursor (in bytes) |
| LOADED_VERSIONS | NUMBER | | Indicates whether the context heap is loaded (1) or not (0) |
| VERSION_COUNT | NUMBER | | Number of children associated with the cursor |
| MODULE | VARCHAR2(64) | | Contains the name of the module that was executing at the time that the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_MODULE |
| ACTION | VARCHAR2(64) | | Contains the name of the action that was executing at the time that the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_ACTION |
| SQL_PROFILE | VARCHAR2(64) | | Name of the applied SQL Profile |
| FORCE_MATCHING_SIGNATURE | NUMBER | | The signature used when the CURSOR_SHARING parameter is set to FORCE |
| PARSING_SCHEMA_ID | NUMBER | | Schema ID that was used to originally build the child cursor |
| PARSING_SCHEMA_NAME | VARCHAR2(128) | | Schema name that was used to originally build the child cursor |
| PARSING_USER_ID | NUMBER | | User ID that was used to originally build the child cursor |
| FETCHES_TOTAL | NUMBER | | Cumulative number of fetches associated with the SQL statement |
| FETCHES_DELTA | NUMBER | | Delta number of fetches associated with the SQL statement |
| END_OF_FETCH_COUNT_TOTAL | NUMBER | | Cumulative number of times this cursor was fully executed since the cursor was brought into the |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less or equal to the value of the EXECUTIONS column. |
| END_OF_FETCH_COUNT_DELTA | NUMBER | | Delta number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. |
| SORTS_TOTAL | NUMBER | | Cumulative number of sorts that were done for this child cursor |
| SORTS_DELTA | NUMBER | | Delta number of sorts that were done for this child cursor |
| EXECUTIONS_TOTAL | NUMBER | | Cumulative number of executions that took place on this object since it was brought into the library cache |
| EXECUTIONS_DELTA | NUMBER | | Delta number of executions that took place on this object since it was brought into the library cache |
| PX_SERVERS_EXECS_TOTAL | NUMBER | | Cumulative number of PX server executions |
| PX_SERVERS_EXECS_DELTA | NUMBER | | Delta number of PX server executions |
| LOADS_TOTAL | NUMBER | | Cumulative number of times the object was either loaded or reloaded |
| LOADS_DELTA | NUMBER | | Delta number of times the object was either loaded or reloaded |
| INVALIDATIONS_TOTAL | NUMBER | | Cumulative number of times this child cursor has been invalidated |
| INVALIDATIONS_DELTA | NUMBER | | Delta number of times this child cursor has been invalidated |
| PARSE_CALLS_TOTAL | NUMBER | | Cumulative number of parse calls for this child cursor |
| PARSE_CALLS_DELTA | NUMBER | | Delta number of parse calls for this child cursor |
| DISK_READS_TOTAL | NUMBER | | Cumulative number of disk reads for this child cursor |
| DISK_READS_DELTA | NUMBER | | Delta number of disk reads for this child cursor |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BUFFER_GETS_TOTAL | NUMBER | | Cumulative number of buffer gets for this child cursor |
| BUFFER_GETS_DELTA | NUMBER | | Delta number of buffer gets for this child cursor |
| ROWS_PROCESSED_TOTAL | NUMBER | | Cumulative number of rows the parsed SQL statement returns |
| ROWS_PROCESSED_DELTA | NUMBER | | Delta number of rows the parsed SQL statement returns |
| CPU_TIME_TOTAL | NUMBER | | Cumulative value of CPU time (in microseconds) used by this cursor for parsing/executing/fetching |
| CPU_TIME_DELTA | NUMBER | | Delta value of CPU time (in microseconds) used by this cursor for parsing/executing/fetching |
| ELAPSED_TIME_TOTAL | NUMBER | | Cumulative value of elapsed time (in microseconds) used by this cursor for parsing/executing/fetching. If the cursor uses parallel execution, then ELAPSED_TIME_TOTAL is the cumulative time for the query coordinator, plus all parallel query slave processes. |
| ELAPSED_TIME_DELTA | NUMBER | | Delta value of elapsed time (in microseconds) used by this cursor for parsing/executing/fetching |
| IOWAIT_TOTAL | NUMBER | | Cumulative value of user I/O wait time (in microseconds) |
| IOWAIT_DELTA | NUMBER | | Delta value of user I/O wait time (in microseconds) |
| CLWAIT_TOTAL | NUMBER | | Cumulative value of cluster wait time (in microseconds) |
| CLWAIT_DELTA | NUMBER | | Delta value of cluster wait time (in microseconds) |
| APWAIT_TOTAL | NUMBER | | Cumulative value of application wait time (in microseconds) |
| APWAIT_DELTA | NUMBER | | Delta value of application wait time (in microseconds) |
| CCWAIT_TOTAL | NUMBER | | Cumulative value of concurrency wait time (in microseconds) |
| CCWAIT_DELTA | NUMBER | | Delta value of concurrency wait time (in microseconds) |
| DIRECT_WRITES_TOTAL | NUMBER | | Cumulative value of direct writes |
| DIRECT_WRITES_DELTA | NUMBER | | Delta value of direct writes |
| PLSEXEC_TIME_TOTAL | NUMBER | | Cumulative value of PL/SQL Execution Time (in microseconds) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PLSEXEC_TIME_DELTA | NUMBER | | Delta value of PL/SQL Execution Time (in microseconds) |
| JAVEXEC_TIME_TOTAL | NUMBER | | Cumulative value of Java Execution Time (in microseconds) |
| JAVEXEC_TIME_DELTA | NUMBER | | Delta value of Java Execution Time (in microseconds) |
| IO_OFFLOAD_ELIG_BYTES_TOTAL | NUMBER | | Cumulative value of number of I/O bytes which can be filtered by the Exadata storage system<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_OFFLOAD_ELIG_BYTES_DELTA | NUMBER | | Delta value of number of I/O bytes which can be filtered by the Exadata storage system<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_INTERCONNECT_BYTES_TOTAL | NUMBER | | Cumulative value of number of I/O bytes exchanged between Oracle Database and the storage system |
| IO_INTERCONNECT_BYTES_DELTA | NUMBER | | Delta value of number of I/O bytes exchanged between Oracle Database and the storage system |
| PHYSICAL_READ_REQUESTS_TOTAL | NUMBER | | Cumulative value of number of physical read I/O requests issued by the monitored SQL |
| PHYSICAL_READ_REQUESTS_DELTA | NUMBER | | Delta value of number of physical read I/O requests issued by the monitored SQL |
| PHYSICAL_READ_BYTES_TOTAL | NUMBER | | Cumulative value of number of bytes read from disks by the monitored SQL |
| PHYSICAL_READ_BYTES_DELTA | NUMBER | | Delta value of number of bytes read from disks by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS_TOTAL | NUMBER | | Cumulative value of number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS_DELTA | NUMBER | | Delta value of number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_BYTES_TOTAL | NUMBER | | Cumulative value of number of bytes written to disks by the monitored SQL |
| PHYSICAL_WRITE_BYTES_DELTA | NUMBER | | Delta value of number of bytes written to disks by the monitored SQL |
| OPTIMIZED_PHYSICAL_READS_TOTAL | NUMBER | | Cumulative value of number of physical reads from the Database Smart Flash Cache or the Exadata Smart Flash Cache by the monitored SQL |
| OPTIMIZED_PHYSICAL_READS_DELTA | NUMBER | | Delta value of number of physical reads from the Database Smart Flash Cache or the Exadata Smart Flash Cache by the monitored SQL |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CELL_UNCOMPRESSED_BYTES_TOTAL | NUMBER | | Cumulative value of number of uncompressed bytes (that is, size after decompression) that are offloaded to the Exadata cells<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| CELL_UNCOMPRESSED_BYTES_DELTA | NUMBER | | Delta value of number of uncompressed bytes (that is, size after decompression) that are offloaded to the Exadata cells<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_OFFLOAD_RETURN_BYTES_TOTAL | NUMBER | | Cumulative value of number of bytes that are returned by the Exadata cell for smart scan only (that is, not including bytes for other database I/O)<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_OFFLOAD_RETURN_BYTES_DELTA | NUMBER | | Delta value of number of bytes that are returned by the Exadata cell for smart scan only (that is, not including bytes for other database I/O)<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| BIND_DATA | RAW(2000) | | Bind data |
| FLAG | NUMBER | | Reserved for internal use |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "DBA_HIST_SNAPSHOT"
- "DBA_HIST_OPTIMIZER_ENV"
- "DBA_HIST_SQLTEXT"
- "DBA_HIST_SQL_PLAN"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO package

# DBA_HIST_SQLTEXT

DBA_HIST_SQLTEXT displays the text of SQL statements belonging to shared SQL cursors captured in the Workload Repository. This view captures information from V$SQL and is used with the DBA_HIST_SQLSTAT view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| SQL_TEXT | CLOB | | Full text for the SQL statement exposed as a CLOB column |
| COMMAND_TYPE | NUMBER | | Oracle command type definition |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SQL"
- "DBA_HIST_SQLSTAT"

# DBA_HIST_STAT_NAME

DBA_HIST_STAT_NAME displays decoded statistic names for the statistics captured in the Workload Repository. This includes OLAP statisitics and OLAP timed events. This view captures information from V$STATNAME and is used with DBA_HIST_SYSSTAT and DBA_HIST_SYS_TIME_MODEL.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| STAT_ID | NUMBER | NOT NULL | Statistic identifier |
| STAT_NAME | VARCHAR2(64) | NOT NULL | Statistic name |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$STATNAME"
- "DBA_HIST_SYSSTAT"
- "DBA_HIST_SYS_TIME_MODEL"

# DBA_HIST_STREAMS_APPLY_SUM

DBA_HIST_STREAMS_APPLY_SUM displays information about each apply process and its activities. This view contains a snapshot of V$STREAMS_APPLY_COORDINATOR, V$STREAMS_APPLY_READER, and V$STREAMS_APPLY_SERVER. This view is intended for use with Automatic Workload Repository (AWR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| APPLY_NAME | VARCHAR2(128) | NOT NULL | Name of the apply process |
| STARTUP_TIME | DATE | NOT NULL | Time that the apply process was last started |
| READER_TOTAL_MESSAGES_DEQUEUED | NUMBER | | Total number of messages dequeued since the apply process was last started |
| READER_LAG | NUMBER | | For captured messages, the delay (in seconds) between the creation of the last message and it being received by the apply process. For user enqueued messages, the delay between the message being enqueued in the local database and being received by the apply process. |
| COORD_TOTAL_RECEIVED | NUMBER | | Total number of transactions received by the coordinator process since the apply process was last started |
| COORD_TOTAL_APPLIED | NUMBER | | Total number of transactions applied by the apply process since the apply process was last started |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COORD_TOTAL_ROLLBACKS | NUMBER | | Number of transactions which were rolled back due to unexpected contention |
| COORD_TOTAL_WAIT_DEPS | NUMBER | | Number of times since the apply process was last started that an apply server waited to apply a logical change record (LCR) in a transaction until another apply server applied a transaction because of a dependency between the transactions |
| COORD_TOTAL_WAIT_CMTS | NUMBER | | Number of times since the apply process was last started that an apply server waited to commit a transaction until another apply server committed a transaction to serialize commits |
| COORD_LWM_LAG | NUMBER | | For captured messages, the delay (in seconds) between the creation of the message corresponding to the low watermark and it being applied by the apply process. For user enqueued messages, the delay between the message being enqueued in the local database and being applied by the apply process. |
| SERVER_TOTAL_MESSAGES_APPLIED | NUMBER | | Total number of messages applied by all the apply servers since the apply process was last started |
| SERVER_ELAPSED_DEQUEUE_TIME | NUMBER | | Time elapsed (in hundredths of a second) dequeuing messages by all the apply servers since the apply process was last started |
| SERVER_ELAPSED_APPLY_TIME | NUMBER | | Time elapsed (in hundredths of a second) applying messages by all the apply servers since the apply process was last started |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$STREAMS_APPLY_COORDINATOR"
- "V$STREAMS_APPLY_READER"
- "V$STREAMS_APPLY_SERVER"

# DBA_HIST_STREAMS_CAPTURE

DBA_HIST_STREAMS_CAPTURE displays information about each capture process. This view contains a snapshot of V$STREAMS_CAPTURE. This view is intended for use with Automatic Workload Repository (AWR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the capture process |
| STARTUP_TIME | DATE | NOT NULL | Time that the capture process was last started |
| LAG | NUMBER | | Delay (in seconds) between the creation and capture of the most recently captured message |
| TOTAL_MESSAGES_CAPTURED | NUMBER | | Total changes captured since the capture process was last started |
| TOTAL_MESSAGES_ENQUEUED | NUMBER | | Total number of messages enqueued since the capture process was last started |
| ELAPSED_RULE_TIME | NUMBER | | Elapsed time (in hundredths of a second) evaluating rules since the capture process was last started |
| ELAPSED_ENQUEUE_TIME | NUMBER | | Elapsed time (in hundredths of a second) enqueuing messages since the capture process was last started |
| ELAPSED_REDO_WAIT_TIME | NUMBER | | Elapsed time (in hundredths of a second) spent by the capture process in the WAITING FOR REDO state |
| ELAPSED_PAUSE_TIME | NUMBER | | Elapsed pause time |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$STREAMS_CAPTURE"

# DBA_HIST_STREAMS_POOL_ADVICE

DBA_HIST_STREAMS_POOL_ADVICE displays historical information about the estimated count of spilled or unspilled messages and the associated time spent in the spill or unspill activity for different Streams pool sizes. This view is intended for use with Automatic Workload Repository (AWR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID of the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number of the snapshot |
| SIZE_FOR_ESTIMATE | NUMBER | NOT NULL | Pool size for the estimate (in megabytes) |
| SIZE_FACTOR | NUMBER | | Size factor with respect to the current pool size |
| ESTD_SPILL_COUNT | NUMBER | | Estimated count of messages spilled from the Streams pool |
| ESTD_SPILL_TIME | NUMBER | | Estimated elapsed time (in seconds) to spill |
| ESTD_UNSPILL_COUNT | NUMBER | | Estimated count of unspills (read back from disk) |
| ESTD_UNSPILL_TIME | NUMBER | | Estimated elapsed time (in seconds) to unspill |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_SYS_TIME_MODEL

DBA_HIST_SYS_TIME_MODEL displays historical system time model statistics, including OLAP timed stastistics. This view contains snapshots of V $SYS_TIME_MODEL.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| STAT_ID | NUMBER | NOT NULL | Statistic ID |
| STAT_NAME | VARCHAR2(64) | NOT NULL | Statistic name |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| VALUE | NUMBER | | Statistic value |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYS_TIME_MODEL"

# DBA_HIST_SYSMETRIC_HISTORY

DBA_HIST_SYSMETRIC_HISTORY externalizes all available history of the system metric values for the entire set of data kept in the database. This view contains snapshots of V$SYSMETRIC_HISTORY.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| BEGIN_TIME | DATE | NOT NULL | Begin time of the interval |
| END_TIME | DATE | NOT NULL | End time of the interval |
| INTSIZE | NUMBER | NOT NULL | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | NOT NULL | Group ID |
| METRIC_ID | NUMBER | NOT NULL | Metric ID |
| METRIC_NAME | VARCHAR2(64) | NOT NULL | Metric name |
| VALUE | NUMBER | NOT NULL | Metric Value |
| METRIC_UNIT | VARCHAR2(64) | NOT NULL | Unit of measurement |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYSMETRIC_HISTORY"

# DBA_HIST_SYSMETRIC_SUMMARY

DBA_HIST_SYSMETRIC_SUMMARY displays a history of statistical summary of all metric values in the System Metrics Long Duration group. This view contains snapshots of V$SYSMETRIC_SUMMARY.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| BEGIN_TIME | DATE | NOT NULL | Begin time of the interval |
| END_TIME | DATE | NOT NULL | End time of the interval |
| INTSIZE | NUMBER | NOT NULL | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | NOT NULL | Group ID |
| METRIC_ID | NUMBER | NOT NULL | Metric ID |
| METRIC_NAME | VARCHAR2(64) | NOT NULL | Metric name |
| METRIC_UNIT | VARCHAR2(64) | NOT NULL | Unit of measurement |
| NUM_INTERVAL | NUMBER | NOT NULL | Number of intervals observed |
| MINVAL | NUMBER | NOT NULL | Minimum value observed |
| MAXVAL | NUMBER | NOT NULL | Maximum value observed |
| AVERAGE | NUMBER | NOT NULL | Average over the period |
| STANDARD_DEVIATION | NUMBER | NOT NULL | One standard deviation |
| SUM_SQUARES | NUMBER | | Sum of the squared deviations from the mean |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYSMETRIC_SUMMARY"

# DBA_HIST_SYSSTAT

DBA_HIST_SYSSTAT displays historical system statistics information, including OLAP kernel statistics. This view contains snapshots of V$SYSSTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| STAT_ID | NUMBER | NOT NULL | Statistic identifier |
| STAT_NAME | VARCHAR2(64) | NOT NULL | Statistic name |
| VALUE | NUMBER | | Statistic value |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYSSTAT"

# DBA_HIST_SYSTEM_EVENT

DBA_HIST_SYSTEM_EVENT displays historical information on total waits for an event. This view contains snapshots of V$SYSTEM_EVENT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| EVENT_ID | NUMBER | NOT NULL | Identifier of the wait event |
| EVENT_NAME | VARCHAR2(64) | NOT NULL | Name of the wait event |
| WAIT_CLASS_ID | NUMBER | | Identifier of the Class of the Wait Event |
| WAIT_CLASS | VARCHAR2(64) | | Name of the Class of the Wait Event |
| TOTAL_WAITS | NUMBER | | Total number of waits for the event |
| TOTAL_TIMEOUTS | NUMBER | | Total number of timeouts for the event |
| TIME_WAITED_MICRO | NUMBER | | Total amount of time waited for the event (in microseconds) |
| TOTAL_WAITS_FG | NUMBER | | Total number of waits for the event, from foreground sessions |
| TOTAL_TIMEOUTS_FG | NUMBER | | Total number of timeouts for the event, from foreground sessions |
| TIME_WAITED_MICRO_FG | NUMBER | | Amount of time waited for the event (in microseconds), from foreground sessions |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYSTEM_EVENT"

# DBA_HIST_TABLESPACE

DBA_HIST_TABLESPACE displays tablespace information contained in the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TS# | NUMBER | NOT NULL | Tablespace number |
| TSNAME | VARCHAR2(30) | NOT NULL | Tablespace name |
| CONTENTS | VARCHAR2(30) | | Tablespace contents:<br>• UNDO<br>• PERMANENT<br>• TEMPORARY |
| SEGMENT_SPACE_MANAGEMENT | VARCHAR2(30) | | Indicates whether the free and used space in the tablespace is managed using free lists (MANUAL) or bitmaps (AUTO) |
| EXTENT_MANAGEMENT | VARCHAR2(30) | | Indicates whether the extents in the tablespace are dictionary managed (DICTIONARY) or locally managed (LOCAL) |
| BLOCK_SIZE | NUMBER | | Block size of the tablespace |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_TABLESPACE_STAT

DBA_HIST_TABLESPACE_STAT displays tablespace information from the control file. This view contains snapshots of V$TABLESPACE and DBA_TABLESPACES.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| TS# | NUMBER | NOT NULL | Tablespace number |
| TSNAME | VARCHAR2(30) | | Tablespace name |
| CONTENTS | VARCHAR2(9) | | Tablespace contents:<br>• PERMANENT<br>• TEMPORARY |
| STATUS | VARCHAR2(9) | | Tablespace status:<br>• ONLINE<br>• OFFLINE<br>• READ ONLY |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEGMENT_SPACE_MANAGEMENT | VARCHAR2(6) | | Indicates whether the free and used segment space in the tablespace is managed using free lists (MANUAL) or bitmaps (AUTO) |
| EXTENT_MANAGEMENT | VARCHAR2(10) | | Indicates whether the extents in the tablespace are dictionary managed (DICTIONARY) or locally managed (LOCAL) |
| IS_BACKUP | VARCHAR2(5) | | Indicates whether the tablespace is part of a backup |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$TABLESPACE"
- "DBA_TABLESPACES"

# DBA_HIST_TBSPC_SPACE_USAGE

DBA_HIST_TBSPC_SPACE_USAGE displays historical tablespace usage statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| TABLESPACE_ID | NUMBER | | Tablespace ID |
| TABLESPACE_SIZE | NUMBER | | Tablespace size (in database blocks) |
| TABLESPACE_MAXSIZE | NUMBER | | Maximum size of the tablespace (in database blocks) |
| TABLESPACE_USEDSIZE | NUMBER | | Used size of the tablespace (in database blocks) |
| RTIME | VARCHAR2(25) | | Runtime |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_TEMPFILE

DBA_HIST_TEMPFILE displays a history of the temp file information from the control file. This view contains snapshots of V$TEMPFILE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| FILE# | NUMBER | NOT NULL | File identification number |
| CREATION_CHANGE# | NUMBER | NOT NULL | Change number at which the temp file was created |
| FILENAME | VARCHAR2(513) | NOT NULL | Name of the temp file |
| TS# | NUMBER | NOT NULL | Tablespace number |
| TSNAME | VARCHAR2(30) | | Name of the tablespace |
| BLOCK_SIZE | NUMBER | | Block size of the temp file |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$TEMPFILE"

# DBA_HIST_TEMPSTATXS

DBA_HIST_TEMPSTATXS displays information about temporary file read/write statistics. This view contains snapshots of V$TEMPSTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| FILE# | NUMBER | NOT NULL | File identification number |
| CREATION_CHANGE# | NUMBER | NOT NULL | Change number at which the temp file was created |
| FILENAME | VARCHAR2(513) | NOT NULL | Name of the temp file |
| TS# | NUMBER | NOT NULL | Tablespace number |
| TSNAME | VARCHAR2(30) | | Name of the tablespace |
| BLOCK_SIZE | NUMBER | | Block size of the temp file |
| PHYRDS | NUMBER | | Number of physical reads done |
| PHYWRTS | NUMBER | | Number of times DBWR is required to write |
| SINGLEBLKRDS | NUMBER | | Number of single block reads |
| READTIM | NUMBER | | Time (in hundredths of a second) spent doing reads if the TIMED_STATISTICS parameter is true; 0 if false |
| WRITETIM | NUMBER | | Time (in hundredths of a second) spent doing writes if the TIMED_STATISTICS parameter is true; 0 if false |
| SINGLEBLKRDTIM | NUMBER | | Cumulative single block read time (in hundredths of a second) |
| PHYBLKRD | NUMBER | | Number of physical blocks read |
| PHYBLKWRT | NUMBER | | Number of blocks written to disk, which may be the same as PHYWRTS if all writes are single blocks |
| WAIT_COUNT | NUMBER | | Shows the number of waits at the file level for contended buffers. This value includes the individual wait events that are included in the buffer busy waits wait event.<br>**See Also:** "buffer busy waits" |
| TIME | NUMBER | | Time spent waiting for the wait events in the WAIT_COUNT column |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$TEMPSTAT"

# DBA_HIST_THREAD

DBA_HIST_THREAD displays historical thread information from the control file.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| THREAD# | NUMBER | NOT NULL | Thread number |
| THREAD_INSTANCE_NUMBER | NUMBER | | Instance number of the thread |
| STATUS | VARCHAR2(6) | | Thread status (OPEN) or (CLOSED) |
| OPEN_TIME | DATE | | Last time the thread was opened |
| CURRENT_GROUP# | NUMBER | | Current log group |
| SEQUENCE# | NUMBER | | Sequence number of the current log |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_TOPLEVELCALL_NAME

DBA_HIST_TOPLEVELCALL_NAME displays the mapping between Oracle top level calls and names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TOP_LEVEL_CALL# | NUMBER | NOT NULL | Oracle top level call number |
| TOP_LEVEL_CALL_NAME | VARCHAR2(64) | | Oracle top level call name |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_UNDOSTAT

DBA_HIST_UNDOSTAT displays the history of histograms of statistical data to show how well the system is working. The available statistics include undo space consumption, transaction concurrency, and length of queries executed in the instance. This view contains snapshots of V$UNDOSTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BEGIN_TIME | DATE | NOT NULL | Identifies the beginning of the time interval |
| END_TIME | DATE | NOT NULL | Identifies the end of the time interval |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| UNDOTSN | NUMBER | NOT NULL | Represents the last active undo tablespace in the duration of time. The tablespace ID of the active undo tablespace is returned in this column. If more than one undo tablespace was active in that period, the active undo tablespace that was active at the end of the period is reported. |
| UNDOBLKS | NUMBER | | Represents the total number of undo blocks consumed. You can use this column to obtain the consumption rate of undo blocks, and thereby estimate the size of the undo tablespace needed to handle the workload on your system. |
| TXNCOUNT | NUMBER | | Identifies the total number of transactions executed within the period |
| MAXQUERYLEN | NUMBER | | Identifies the length of the longest query (in number of seconds) executed in the instance during the period. You can use this statistic to estimate the proper setting of the UNDO_RETENTION initialization parameter. The |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | length of a query is measured from the cursor open time to the last fetch/execute time of the cursor. Only the length of those cursors that have been fetched/executed during the period are reflected in the view. |
| MAXQUERYSQLID | VARCHAR2(13) | | SQL identifier of the longest running SQL statement in the period |
| MAXCONCURRENCY | NUMBER | | Identifies the highest number of transactions executed concurrently within the period |
| UNXPSTEALCNT | NUMBER | | Number of attempts to obtain undo space by stealing unexpired extents from other transactions |
| UNXPBLKRELCNT | NUMBER | | Number of unexpired blocks removed from certain undo segments so they can be used by other transactions |
| UNXPBLKREUCNT | NUMBER | | Number of unexpired undo blocks reused by transactions |
| EXPSTEALCNT | NUMBER | | Number of attempts to steal expired undo blocks from other undo segments |
| EXPBLKRELCNT | NUMBER | | Number of expired undo blocks stolen from other undo segments |
| EXPBLKREUCNT | NUMBER | | Number of expired undo blocks reused within the same undo segments |
| SSOLDERRCNT | NUMBER | | Identifies the number of times the error ORA-01555 occurred. You can use this statistic to decide whether the UNDO_RETENTION initialization parameter is set properly given the size of the undo tablespace. Increasing the value of UNDO_RETENTION can reduce the occurrence of this error. |
| NOSPACEERRCNT | NUMBER | | Identifies the number of times space was requested in the undo tablespace and there was no free space available. That is, all of the space in the undo tablespace was in use by active transactions. The corrective action is to add more space to the undo tablespace. |
| ACTIVEBLKS | NUMBER | | Total number of blocks in the active extents of the undo tablespace for the instance at the sampled time in the period |
| UNEXPIREDBLKS | NUMBER | | Total number of blocks in the unexpired extents of the undo tablespace for the instance at the sampled time in the period |
| EXPIREDBLKS | NUMBER | | Total number of blocks in the expired extents of the undo tablespace for the instance at the sampled time in the period |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TUNED_UNDORETENTION | NUMBER | | System tuned value indicating the period for which undo is being retained |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$UNDOSTAT"

# DBA_HIST_WAITCLASSMET_HISTORY

DBA_HIST_WAITCLASSMET_HISTORY displays the history of the wait event class metric data kept by the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID of the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number of the snapshot |
| WAIT_CLASS_ID | NUMBER | NOT NULL | Identifier of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | | Name of the class of the wait event |
| BEGIN_TIME | DATE | NOT NULL | Begin time of the interval |
| END_TIME | DATE | NOT NULL | End time of the interval |
| INTSIZE | NUMBER | NOT NULL | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | NOT NULL | Metric group ID |
| AVERAGE_WAITER_COUNT | NUMBER | NOT NULL | Average waiter count |
| DBTIME_IN_WAIT | NUMBER | NOT NULL | Percent of database time spent in the wait |
| TIME_WAITED | NUMBER | NOT NULL | Time waited during the interval (in microseconds) |
| WAIT_COUNT | NUMBER | NOT NULL | Number of times waited |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TIME_WAITED_FG | NUMBER | | Time waited (in hundredths of a second), from foreground sessions |
| WAIT_COUNT_FG | NUMBER | | Number of times waited, from foreground sessions |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HIST_WAITSTAT

DBA_HIST_WAITSTAT displays historical block contention statistics. This view contains snapshots of V$WAITSTAT.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAP_ID | NUMBER | NOT NULL | Unique snapshot ID |
| DBID | NUMBER | NOT NULL | Database ID for the snapshot |
| INSTANCE_NUMBER | NUMBER | NOT NULL | Instance number for the snapshot |
| CLASS | VARCHAR2(18) | NOT NULL | Class of the block |
| WAIT_COUNT | NUMBER | | Number of waits by the OPERATION for this CLASS of block |
| TIME | NUMBER | | Sum of all wait times for all the waits by the OPERATION for this CLASS of block |
| CON_DBID | NUMBER | | The database ID of the PDB for the sampled session |
| CON_ID | NUMBER | | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$WAITSTAT"

# DBA_HIST_WR_CONTROL

DBA_HIST_WR_CONTROL displays the control information for the Workload Repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBID | NUMBER | NOT NULL | Database ID |
| SNAP_INTERVAL | INTERVAL DAY(5) TO SECOND(1) | NOT NULL | Snapshot interval; how often to automatically take snapshots |
| RETENTION | INTERVAL DAY(5) TO SECOND(1) | NOT NULL | Retention setting for the snapshots; amount of time to keep the snapshots |
| TOPNSQL | VARCHAR2(10) | | The number of Top SQL flushed for each SQL criteria (elapsed time, CPU time, parse calls, sharable memory, version count) |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_HISTOGRAMS

DBA_HISTOGRAMS is a synonym for DBA_TAB_HISTOGRAMS.

**See Also:**

"DBA_TAB_HISTOGRAMS"

# DBA_HOST_ACES

DBA_HOST_ACES describes access control entries defined in host access control lists.

### Related View

USER_HOST_ACES describes the status of access control entries for the current user to access network hosts through PL/SQL network utility packages. This view does not display the ACE_ORDER, START_DATE, END_DATE, GRANT_TYPE, INVERTED_PRINCIPAL, PRINCIPAL, or PRINCIPAL_TYPE columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HOST | VARCHAR2(1000) | NOT NULL | Network host |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOWER_PORT | NUMBER(5) | | Lower bound of the port range |
| UPPER_PORT | NUMBER(5) | | Upper bound of the port range |
| ACE_ORDER | NUMBER | NOT NULL | Order number of the access control entry |
| START_DATE | TIMESTAMP(6) | | Start date of the access control entry |
| END_DATE | TIMESTAMP(6) | | End date of the access control entry |
| GRANT_TYPE | VARCHAR2(5) | | Indicates whether the access control entry grants or denies the privilege |
| INVERTED_PRINCIPAL | VARCHAR2(3) | | Indicates whether the principal is inverted or not |
| PRINCIPAL | VARCHAR2(128) | | Principal the privilege is applied to |
| PRINCIPAL_TYPE | VARCHAR2(16) | | Type of the principal |
| PRIVILEGE | VARCHAR2(128) | | Privilege |

**See Also:**

"USER_HOST_ACES"

# DBA_HOST_ACLS

DBA_HOST_ACLS describes access control lists assigned to restrict access to network hosts through PL/SQL network utility packages.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HOST | VARCHAR2(1000) | NOT NULL | Network host |
| LOWER_PORT | NUMBER(5) | | Lower bound of the port range |
| UPPER_PORT | NUMBER(5) | | Upper bound of the port range |
| ACL | VARCHAR2(4000) | | The name of the access control list |
| ACLID | RAW(8) | | The object ID of the access control list |
| ACL_OWNER | VARCHAR2(128) | | The owner of the access control list |

# DBA_IDENTIFIERS

DBA_IDENTIFIERS displays information about the identifiers in all stored objects in the database. Its columns are the same as those in ALL_IDENTIFIERS.

**See Also:**

"ALL_IDENTIFIERS"

# DBA_ILMDATAMOVEMENTPOLICIES

DBA_ILMDATAMOVEMENTPOLICIES contains information specific to data movement-related attributes of an Automatic Data Optimization policy in a database.

### Related View

USER_ILMDATAMOVEMENTPOLICIES contains information specific to data movement-related attributes of an Automatic Data Optimization policy for a user.

**Note:**

Automatic Data Optimization is not supported with a CDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| POLICY_NAME | VARCHAR2(128) | NOT NULL | The Automatic Data Optimization policy name is autogenerated |
| ACTION_TYPE | VARCHAR2(11) | | Type of data movement action performed by the Automatic Data Optimization policy:<br>• COMPRESSION<br>• STORAGE |
| SCOPE | VARCHAR2(7) | | Identifies the scope of the Automatic Data Optimization policy:<br>• ROW<br>• GROUP<br>• SEGMENT |
| COMPRESSION_LEVEL | VARCHAR2(30) | | Compression level to move selected rows or the entire segment to, for a compression Automatic Data Optimization policy |
| TIER_TABLESPACE | VARCHAR2(128) | | Tablespace to move the object to, for a storage Automatic Data Optimization policy |
| TIER_STATUS | VARCHAR2(9) | | This column is only valid for storage tiering policies, and indicates whether the storage tiering policy was specified with a READ ONLY clause. This column takes READ ONLY as a potential value. In all other cases, it is blank.<br>In other words, this column indicates whether the tablespace the object is being moved to will be made READ ONLY after the movement. |
| CONDITION_TYPE | VARCHAR2(22) | | Column on which the condition for the policy is based. Possible values:<br>• LAST ACCESS TIME<br>• LAST MODIFICATION TIME<br>• CREATION TIME<br>• USER DEFINED |
| CONDITION_DAYS | NUMBER | | Number of days in the condition for the policy |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CUSTOM_FUNCTION | VARCHAR2(128) | | Optional function that evaluates the precondition on the policy |

**See Also:**

"USER_ILMDATAMOVEMENTPOLICIES"

# DBA_ILMEVALUATIONDETAILS

DBA_ILMEVALUATIONDETAILS displays details on evaluation of Automatic Data Optimization policies considered for Automatic Data Optimization tasks. It also shows the job name that executes the policy, in case the policy was selected for execution. If the policy was not selected for execution, this view provides a reason.

### Related View

USER_ILMEVALUATIONDETAILS displays details on evaluation of Automatic Data Optimization policies considered for Automatic Data Optimization tasks for a user. It also shows the job name that executes the policy, in case the policy was selected for execution. If the policy was not selected for execution, this view provides a reason.

**Note:**

Automatic Data Optimization is not supported with a CDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER | | Number that uniquely identifies a specific Automatic Data Optimization task |
| POLICY_NAME | VARCHAR2(128) | NOT NULL | Name of the Automatic Data Optimization policy |
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object associated with the Automatic Data Optimization policy |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object associated with the Automatic Data Optimization policy |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject associated with the Automatic Data Optimization policy |
| OBJECT_TYPE | VARCHAR2(18) | | Object type. Valid values include TABLE, TABLE PARTITION, and TABLE SUBPARTITION. |
| SELECTED_FOR_EXECUTION | VARCHAR2(42) | | Indicates whether the policy has been selected for execution on the object. If not, the reason for not being selected is listed. Possible values:<br>• POLICY DISABLED<br>• SELECTED FOR EXECUTION<br>• POLICY OVERRULED<br>• INHERITED POLICY OVERRULED<br>• PRECONDITION NOT SATISFIED |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • JOB ALREADY EXISTS<br>• NO OPERATION SINCE LAST ILM ACTION<br>• TABLE HAS MATERIALIZED VIEW<br>• TARGET COMPRESSION NOT HIGHER THAN CURRENT<br>• STATISTICS NOT AVAILABLE<br>The value SELECTED FOR EXECUTION means a job was created for this policy on the object. The other values state the reason why the policy on the object was selected for execution. |
| JOB_NAME | VARCHAR2(128) | | Name of the job in the case where the policy is selected for execution on this object |
| COMMENTS | VARCHAR2(4000) | | Reserved for future use |

**See Also:**

"USER_ILMEVALUATIONDETAILS"

# DBA_ILMOBJECTS

DBA_ILMOBJECTS displays all the Automatic Data Optimization policies and objects in the database. Many objects inherit policies via their parent objects or because they were created in a particular tablespace. This view provides a mapping between the policies and objects and indicates whether a policy is inherited by an object or is directly specified on it.

### Related View

USER_ILMOBJECTS displays all the Automatic Data Optimization policies and objects for a user.

**Note:**

Automatic Data Optimization is not supported with a CDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| POLICY_NAME | VARCHAR2(128) | NOT NULL | Policy name is auto-generated |
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object associated with the Automatic Data Optimization policy |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object associated with the Automatic Data Optimization policy |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject associated with the Automatic Data Optimization policy |
| OBJECT_TYPE | VARCHAR2(18) | | Object type. Valid values include:<br>• INDEX |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • INDEX PARTITION<br>• LOB<br>• LOB PARTITION<br>• TABLE<br>• TABLE PARTITION<br>• TABLE SUBPARTITION<br>Direct policies on Index, Index Partition, LOB, and LOB Partition are not supported in Oracle Database 12*c*. |
| INHERITED_FROM | VARCHAR2(20) | | Indicates if the policy is inherited by the object or subobject, or directly specified on the object or subobject. If the policy is inherited, the level from which the policy is inherited (TABLE, TABLE PARTITION, TABLESPACE) is identified. |
| ENABLED | VARCHAR2(7) | | Indicates if the Automatic Data Optimization policy is enabled for the object (YES or NO) |
| DELETED | VARCHAR2(7) | | Possible values:<br>• YES - Indicates that the policy is deleted for any objects that may be added in the future, but is active for those objects that are currently associated with that policy<br>• NO - Indicates that the policy is active |

**See Also:**

"USER_ILMOBJECTS"

# DBA_ILMPARAMETERS

DBA_ILMPARAMETERS can be queried to provide information on the Automatic Data Optimization parameters in the database and their values.

**Note:**

Automatic Data Optimization is not supported with a CDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | | Name of the Automatic Data Optimization environment parameter. The value is one of the constants defined in the DBMS_ILM_ADMIN package. |
| VALUE | NUMBER | | Value of the parameter |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the API interface for implementing Automatic Data Optimization strategies
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_ILM_ADMIN` package

# DBA_ILMPOLICIES

`DBA_ILMPOLICIES` displays details about Automatic Data Optimization policies in the database. The view contains common details relevant to all types of Automatic Data Optimization policies, not just details relevant to the data movement-related Automatic Data Optimization policies.

### Related View

`USER_ILMPOLICIES` displays details about Automatic Data Optimization policies owned by the user.

**Note:**

Automatic Data Optimization is not supported with a CDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `POLICY_NAME` | `VARCHAR2(128)` | | The name of the Automatic Data Optimization policy is auto-generated |
| `POLICY_TYPE` | `VARCHAR2(13)` | | Type of the policy. Valid values include `DATAMOVEMENT`. |
| `TABLESPACE` | `VARCHAR2(30)` | | Tablespace name, in the case of a tablespace-level policy |
| `ENABLED` | `VARCHAR2(6)` | | Indicates whether the policy is enabled or not |
| `DELETED` | `VARCHAR2(7)` | | Possible values:<br>• `YES` - Indicates that the policy associated has been deleted (but shall remain active for this object)<br>• `NO` - Indicates that the policy is active |

**Note:**

Automatic Data Optimization policies cannot be set on tables with object types or materialized views.

**See Also:**

"USER_ILMPOLICIES"

# DBA_ILMRESULTS

DBA_ILMRESULTS displays information on data movement-related Automatic Data Optimization jobs in the database.

### Related View

USER_ILMRESULTS displays information on data movement-related Automatic Data Optimization jobs for tasks created by the user.

**Note:**

Automatic Data Optimization is not supported with a CDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER | | Number that uniquely identifies a specific Automatic Data Optimization task |
| JOB_NAME | VARCHAR2(128) | | Name of the Automatic Data Optimization job |
| JOB_STATE | VARCHAR2(31) | | State of the job. Possible values:<br>• JOB CREATED<br>• COMPLETED SUCCESSFULLY<br>• FAILED<br>• STOPPED<br>• JOB CREATION FAILED<br>• DEPENDENT OBJECTS BEING REBUILT<br>• FAILED TO REBUILD DEPENDENT OBJECTS |
| START_TIME | TIMESTAMP(6) | | Start time of the Automatic Data Optimization job |
| COMPLETION_TIME | TIMESTAMP(6) | | Completion time of the Automatic Data Optimization job |
| COMMENTS | VARCHAR2(300) | | Additional information in cases where the Automatic Data Optimization job execution fails |
| STATISTICS | CLOB | | Job specific statistics, such as space saved via compression. This column is in the form of comma separated name / value pairs, with each pair representing a particular statistic name and value. |

**See Also:**

"USER_ILMRESULTS"

# DBA_ILMTASKS

DBA_ILMTASKS displays information on Automatic Data Optimization execution.

**Related View**

`USER_ILMTASKS` displays information on Automatic Data Optimization tasks created by a user.

**Note:**

Automatic Data Optimization is not supported with a CDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `TASK_ID` | `NUMBER` | | Number that uniquely identifies a specific Automatic Data Optimization task |
| `TASK_OWNER` | `VARCHAR2(128)` | `NOT NULL` | User who initiates the task |
| `STATE` | `VARCHAR2(9)` | | Possible values:<br>• `INACTIVE`: Indicates that the task was created for previewing<br>• `ACTIVE`: Indicates that jobs have been created for the qualifying policies in the task<br>• `COMPLETE`: Indicates that the task has completed |
| `CREATION_TIME` | `TIMESTAMP(6)` | | The time that the task was created |
| `START_TIME` | `TIMESTAMP(6)` | | Start time of a specific task |
| `COMPLETION_TIME` | `TIMESTAMP(6)` | | Completion time of a specific task |

**See Also:**

"USER_ILMTASKS"

# DBA_IND_COLUMNS

`DBA_IND_COLUMNS` describes the columns of all the indexes on all tables and clusters in the database. Its columns are the same as those in "ALL_IND_COLUMNS".

# DBA_IND_EXPRESSIONS

`DBA_IND_EXPRESSIONS` lists expressions of function-based indexes on all tables and clusters in the database. Its columns are the same as those in "ALL_IND_EXPRESSIONS".

# DBA_IND_PARTITIONS

`DBA_IND_PARTITIONS` displays, for each index partition in the database, the partition-level partitioning information, the storage parameters for the partition, and various partition statistics generated by the `DBMS_STATS` package. Its columns are the same as those in "ALL_IND_PARTITIONS".

## DBA_IND_PENDING_STATS

DBA_IND_PENDING_STATS describes pending statistics for all tables, partitions, and subpartitions in the database collected using the DBMS_STATS package. Its columns are the same as those in "ALL_IND_PENDING_STATS".

## DBA_IND_STATISTICS

DBA_IND_STATISTICS displays optimizer statistics for all indexes in the database collected using the DBMS_STATS package. Its columns are the same as those in ALL_IND_STATISTICS.

**See Also:**

"ALL_IND_STATISTICS"

## DBA_IND_SUBPARTITIONS

DBA_IND_SUBPARTITIONS displays, for each index subpartition in the database, the subpartition-level partitioning information, the storage parameters for the subpartition, and various subpartition statistics generated by the DBMS_STATS package. Its columns are the same as those in "ALL_IND_SUBPARTITIONS".

## DBA_INDEXES

DBA_INDEXES describes all indexes in the database. To gather statistics for this view, use the DBMS_STATS package. This view supports parallel partitioned index scans. Its columns are the same as those in "ALL_INDEXES".

## DBA_INDEXTYPE_ARRAYTYPES

DBA_INDEXTYPE_ARRAYTYPES displays information about the array types specified by all indextypes in the database. Its columns are the same as those in ALL_INDEXTYPE_ARRAYTYPES.

**See Also:**

"ALL_INDEXTYPE_ARRAYTYPES"

## DBA_INDEXTYPE_COMMENTS

DBA_INDEXTYPE_COMMENTS displays comments for all user-defined indextypes in the database. Its columns are the same as those in ALL_INDEXTYPE_COMMENTS.

**See Also:**

"ALL_INDEXTYPE_COMMENTS"

## DBA_INDEXTYPE_OPERATORS

DBA_INDEXTYPE_OPERATORS lists all the operators supported by indextypes in the database. Its columns are the same as those in ALL_INDEXTYPE_OPERATORS.

**See Also:**

"ALL_INDEXTYPE_OPERATORS"

## DBA_INDEXTYPES

DBA_INDEXTYPES describes all indextypes in the database. Its columns are the same as those in ALL_INDEXTYPES.

**See Also:**

"ALL_INDEXTYPES"

## DBA_INTERNAL_TRIGGERS

DBA_INTERNAL_TRIGGERS describes internal triggers on all tables in the database. Its columns are the same as those in "ALL_INTERNAL_TRIGGERS".

## DBA_JAVA_ARGUMENTS

DBA_JAVA_ARGUMENTS displays argument information about all stored Java classes in the database. Its columns are the same as those in ALL_JAVA_ARGUMENTS.

**See Also:**

"ALL_JAVA_ARGUMENTS"

## DBA_JAVA_CLASSES

DBA_JAVA_CLASSES displays class level information about all stored Java classes in the database. Its columns are the same as those in ALL_JAVA_CLASSES.

**See Also:**

"ALL_JAVA_CLASSES"

## DBA_JAVA_COMPILER_OPTIONS

DBA_JAVA_COMPILER_OPTIONS displays information about all native compiler options in the database. Its columns are the same as those in ALL_JAVA_COMPILER_OPTIONS.

**See Also:**

"ALL_JAVA_COMPILER_OPTIONS"

## DBA_JAVA_DERIVATIONS

DBA_JAVA_DERIVATIONS displays mapping information about Java source objects and their derived Java class objects and Java resource objects for all Java classes in the database. Its columns are the same as those in ALL_JAVA_DERIVATIONS.

**See Also:**

"ALL_JAVA_DERIVATIONS"

## DBA_JAVA_FIELDS

DBA_JAVA_FIELDS displays field information about all stored Java classes in the database. Its columns are the same as those in ALL_JAVA_FIELDS.

**See Also:**

"ALL_JAVA_FIELDS"

## DBA_JAVA_IMPLEMENTS

DBA_JAVA_IMPLEMENTS describes interfaces implemented by all stored Java classes in the database. Its columns are the same as those in ALL_JAVA_IMPLEMENTS.

**See Also:**

"ALL_JAVA_IMPLEMENTS"

## DBA_JAVA_INNERS

DBA_JAVA_INNERS displays information about inner classes referred to by all stored Java classes in the database. Its columns are the same as those in ALL_JAVA_INNERS.

**See Also:**

"ALL_JAVA_INNERS"

## DBA_JAVA_LAYOUTS

DBA_JAVA_LAYOUTS displays class layout information about all stored Java classes in the database. Its columns are the same as those in ALL_JAVA_LAYOUTS.

**See Also:**

"ALL_JAVA_LAYOUTS"

# DBA_JAVA_METHODS

DBA_JAVA_METHODS displays method information about all stored Java classes in the database. Its columns are the same as those in ALL_JAVA_METHODS.

**See Also:**

"ALL_JAVA_METHODS"

# DBA_JAVA_NCOMPS

DBA_JAVA_NCOMPS displays ncomp-related information about all Java classes in the database. Its columns are the same as those in ALL_JAVA_NCOMPS.

**See Also:**

"ALL_JAVA_NCOMPS"

# DBA_JAVA_POLICY

DBA_JAVA_POLICY describes Java security permissions for all users in the database.

### Related View

USER_JAVA_POLICY describes Java security permissions for the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| KIND | VARCHAR2(8) | | Indicates whether the permission is a positive (GRANT) or a limitation (RESTRICT) |
| GRANTEE | VARCHAR2(128) | NOT NULL | Name of the user, schema, or role to which the permission object is assigned |
| TYPE_SCHEMA | VARCHAR2(128) | NOT NULL | Schema in which the permission object is loaded |
| TYPE_NAME | VARCHAR2(4000) | | Permission class type, which is designated by a string containing the full class name, such as, java.io.FilePermission |
| NAME | VARCHAR2(4000) | | Target attribute (name) of the permission object. This name is used when defining the permission. |
| ACTION | VARCHAR2(4000) | | Action attribute for this permission. Many permissions expect a null value if no action is appropriate for the permission. |
| ENABLED | VARCHAR2(8) | | Indicates whether the permission is enabled (ENABLED) or disabled (DISABLED) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEQ | NUMBER | | Sequence number used to identify this row. This number should be supplied when disabling, enabling, or deleting the permission. |

**See Also:**

"USER_JAVA_POLICY"

# DBA_JAVA_RESOLVERS

DBA_JAVA_RESOLVERS displays information about resolvers of all Java classes in the database. Its columns are the same as those in ALL_JAVA_RESOLVERS.

**See Also:**

"ALL_JAVA_RESOLVERS"

# DBA_JAVA_THROWS

DBA_JAVA_THROWS displays information about exceptions thrown from methods of all Java classes in the database. Its columns are the same as those in ALL_JAVA_THROWSa.

**See Also:**

"ALL_JAVA_THROWS"

# DBA_JOBS

DBA_JOBS describes all jobs in the database.

### Related View

USER_JOBS describes the jobs owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB | NUMBER | NOT NULL | Identifier of job. Neither import/export nor repeated executions change this value. |
| LOG_USER | VARCHAR2(128) | NOT NULL | Login user when the job was submitted |
| PRIV_USER | VARCHAR2(128) | NOT NULL | User whose default privileges apply to this job |
| SCHEMA_USER | VARCHAR2(128) | NOT NULL | Default schema used to parse the job<br>For example, if the SCHEMA_USER is SCOTT and you submit the procedure HIRE_EMP as a job, the Oracle Database looks for SCOTT.HIRE_EMP |
| LAST_DATE | DATE | | Date on which this job last successfully executed |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_SEC | VARCHAR2(8) | | Same as LAST_DATE. This is when the last successful execution started. |
| THIS_DATE | DATE | | Date that this job started executing (usually null if not executing) |
| THIS_SEC | VARCHAR2(8) | | Same as THIS_DATE. This is when the last successful execution started. |
| NEXT_DATE | DATE | NOT NULL | Date that this job will next be executed |
| NEXT_SEC | VARCHAR2(8) | | Same as NEXT_DATE. This is when the last successful execution started. |
| TOTAL_TIME | NUMBER | | Total wall clock time spent by the system on this job (in seconds) since the first time this job executed. This value is cumulative. |
| BROKEN | VARCHAR2(1) | | Y: no attempt is made to run this job<br>N: an attempt is made to run this job |
| INTERVAL | VARCHAR2(200) | NOT NULL | A date function, evaluated at the start of execution, becomes next NEXT_DATE |
| FAILURES | NUMBER | | Number of times the job has started and failed since its last success |
| WHAT | VARCHAR2(4000) | | Body of the anonymous PL/SQL block that the job executes |
| NLS_ENV | VARCHAR2(4000) | | Session parameters describing the NLS environment of the job |
| MISC_ENV | RAW(32) | | Other session parameters that apply to this job |
| INSTANCE | NUMBER | | ID of the instance that can execute or is executing the job. The default is 0. |

**See Also:**

"USER_JOBS"

# DBA_JOBS_RUNNING

DBA_JOBS_RUNNING lists all jobs that are currently running in the instance.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SID | NUMBER | | Identifier of process that is executing the job. See "V$LOCK". |
| JOB | NUMBER | | Identifier of job. This job is currently executing. |
| FAILURES | NUMBER | | Number of times this job started and failed since its last success. |
| LAST_DATE | DATE | | Date that this job last successfully executed. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_SEC | VARCHAR2(8) | | Same as LAST_DATE. This is when the last successful execution started. |
| THIS_DATE | DATE | | Date that this job started executing. |
| THIS_SEC | VARCHAR2(8) | | Same as THIS_DATE. This is when the last successful execution started. |
| INSTANCE | NUMBER | | Indicates which instance can execute or is executing the job; the default is 0. |

# DBA_JOIN_IND_COLUMNS

DBA_JOIN_IND_COLUMNS describes all join conditions in the database. Its columns are the same as those in "ALL_JOIN_IND_COLUMNS" .

# DBA_JSON_COLUMNS

DBA_JSON_COLUMNS provides information on all JavaScript Object Notation (JSON) columns in the database. Its columns are the same as those in ALL_JSON_COLUMNS. Each column that has an IS JSON check constraint in an AND condition appears in this view. This view enables a DBA to find all the JSON columns in the database

For example, if a check constraint combines the IS JSON condition with another condition using logical condition OR, then the column is not listed in this view. In this case, it is not certain that the data in the column is JSON data. For example, the following constraint does not ensure that the data in column jcol is JSON data:

```
jcol is json OR length(jcol) < 1000
```

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_JSON_COLUMNS"
- *Oracle XML DB Developer's Guide* for more information about using JSON with Oracle Database

# DBA_KGLLOCK

DBA_KGLLOCK lists all the locks and pins held on KGL objects (objects in the Kernel Generic Library cache).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| kgllkuse | RAW(4) | | Address of the user session that holds the lock or pin |
| kgllkhdl | RAW(4) | | Address of the handle for the KGL object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| kgllkmod | NUMBER | | Current mode of the lock or pin |
| kgllkreq | NUMBER | | Mode in which the lock or pin was requested |
| kgllktype | VARCHAR2(4) | | Whether this is a lock or a pin |

# DBA_LIBRARIES

DBA_LIBRARIES describes all libraries in the database. Its columns are the same as those in ALL_LIBRARIES.

**See Also:**

"ALL_LIBRARIES"

# DBA_LMT_FREE_SPACE

DBA_LMT_FREE_SPACE describes the free extents in all locally managed tablespaces in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_ID | NUMBER | | Identifier number of the tablespace containing the extent |
| FILE_ID | NUMBER | | File identifier number of the file containing the extent |
| BLOCK_ID | NUMBER | | Starting block number of the extent |
| BLOCKS | NUMBER | | Size of the extent (in Oracle blocks) |

# DBA_LMT_USED_EXTENTS

DBA_LMT_USED_EXTENTS describes the extents comprising the segments in all locally managed tablespaces in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEGMENT_FILEID | NUMBER | | File number of the segment header of the extent |
| SEGMENT_BLOCK | NUMBER | | Block number of the segment header of the extent |
| TABLESPACE_ID | NUMBER | | Identifier number of the tablespace containing the extent |
| EXTENT_ID | NUMBER | | Extent number in the segment |
| FILEID | NUMBER | | File identifier number of the file containing the extent |
| BLOCK | NUMBER | | Starting block number of the extent |
| LENGTH | NUMBER | | Number of blocks in the extent |

## DBA_LOB_PARTITIONS

DBA_LOB_PARTITIONS displays all LOB partitions in the database. Its columns are the same as those in "ALL_LOB_PARTITIONS".

## DBA_LOB_SUBPARTITIONS

DBA_LOB_SUBPARTITIONS displays partition-level attributes of all LOB data subpartitions in the database. Its columns are the same as those in "ALL_LOB_SUBPARTITIONS".

## DBA_LOB_TEMPLATES

DBA_LOB_TEMPLATES describes all LOB subpartition templates in the database. Its columns are the same as those in ALL_LOB_TEMPLATES.

**See Also:**

"ALL_LOB_TEMPLATES"

## DBA_LOBS

DBA_LOBS displays the BLOBs and CLOBs contained in all tables in the database. BFILEs are stored outside the database, so they are not described by this view. This view's columns are the same as those in "ALL_LOBS".

## DBA_LOCK

DBA_LOCK lists all locks or latches held in the database, and all outstanding requests for a lock or latch.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SESSION_ID | NUMBER | | Session holding or acquiring the lock |
| LOCK_TYPE | VARCHAR2(26) | | Lock type<br>**See Also:** For a listing of lock types, see Oracle Enqueue Names |
| MODE HELD | VARCHAR2(40) | | Lock mode |
| MODE REQUESTED | VARCHAR2(40) | | Lock mode requested |
| LOCK_ID1 | VARCHAR2(40) | | Type-specific lock identifier, part 1 |
| LOCK_ID2 | VARCHAR2(40) | | Type-specific lock identifier, part 2 |
| LAST_CONVERT | NUMBER | | The last convert |
| BLOCKING_OTHERS | VARCHAR2(40) | | Whether the lock is currently blocking others |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_LOCK_INTERNAL

DBA_LOCK_INTERNAL displays a row for each lock or latch that is being held, and one row for each outstanding request for a lock or latch.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SESSION_ID | NUMBER | | Session holding or acquiring the lock |
| LOCK_TYPE | VARCHAR2(56) | | Lock type<br>**See Also:** For a listing of lock types, see Oracle Enqueue Names |
| MODE HELD | VARCHAR2(40) | | Lock mode |
| MODE REQUESTED | VARCHAR2(40) | | Lock mode requested |
| LOCK_ID1 | VARCHAR2(1130) | | Type-specific lock identifier, part 1 |
| LOCK_ID2 | VARCHAR2(40) | | Type-specific lock identifier, part 2 |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_LOCKS

DBA_LOCKS is a synonym for DBA_LOCK.

**See Also:**

"DBA_LOCK"

# DBA_LOG_GROUP_COLUMNS

DBA_LOG_GROUP_COLUMNS describes all columns in the database that are specified in log groups. Its columns are the same as those in "ALL_LOG_GROUP_COLUMNS" .

# DBA_LOG_GROUPS

DBA_LOG_GROUPS describes log group definitions on all tables in the database. Its columns are the same as those in "ALL_LOG_GROUPS" .

# DBA_LOGMNR_LOG

DBA_LOGMNR_LOG displays all archived logs registered with active LogMiner persistent sessions in the database.

A persistent LogMiner session is created either by starting Data Guard SQL Apply on a logical standby database for the first time or by creating Streams capture.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOGMNR_SESSION_ID | NUMBER | NOT NULL | Unique identifier of the persistent session |
| NAME | VARCHAR2(513) | | Name of the archived log |
| DBID | NUMBER | NOT NULL | Database identifier that produced the archived log |
| RESETLOGS_SCN | NUMBER | NOT NULL | SCN at which resetlogs operation was performed at the source database generating the archived log |
| RESETLOGS_TIME | NUMBER | NOT NULL | Timestamp at which resetlogs operation was performed at the source database generating the archived log |
| MODIFIED_TIME | DATE | | Time at which the archived log was registered with LogMiner |
| THREAD# | NUMBER | NOT NULL | Redo thread at the source database that generated the archived log |
| SEQUENCE# | NUMBER | NOT NULL | Logfile sequence number |
| FIRST_SCN | NUMBER | NOT NULL | Lowest SCN of the redo record contained in the logfile |
| NEXT_SCN | NUMBER | | Highest possible SCN of the redo record contained in the logfile |
| FIRST_TIME | DATE | | Time of the first redo record contained in the logfile |
| NEXT_TIME | DATE | | Time of the last redo record contained in the logfile |
| DICTIONARY_BEGIN | VARCHAR2(3) | | Indicates whether the archived log contains the beginning of a LogMiner dictionary (YES) or not (NO) |
| DICTIONARY_END | VARCHAR2(3) | | Indicates whether the archived log contains the end of a LogMiner dictionary (YES) or not (NO) |
| KEEP | VARCHAR2(3) | | Indicates whether the logfile is still required for this LogMiner session (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SUSPECT | VARCHAR2(3) | | Indicates whether the archived log content was deemed to be corrupt or the archived log is partially filled (YES) or not (NO) |

# DBA_LOGMNR_PURGED_LOG

DBA_LOGMNR_PURGED_LOG displays archived redo log files that have been applied to the logical standby database and can be deleted because they are no longer needed. FIles in this view are refreshed as a result of executing the DBMS_LOGSTDBY.PURGE_SESSION PL/SQL procedure for Oracle Data Guard SQL Apply:

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_NAME | VARCHAR2(513) | | Fully qualified names of the archived redo log files that are no longer needed by SQL Apply and can be deleted from the operating system |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_LOGSTDBY.PURGE_SESSION procedure

# DBA_LOGMNR_SESSION

DBA_LOGMNR_SESSION displays all active LogMiner persistent sessions in the database.

A persistent LogMiner session is created either by starting Data Guard SQL Apply on a logical standby database for the first time or by creating Streams capture.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ID | NUMBER | NOT NULL | Unique session identifier |
| NAME | VARCHAR2(128) | NOT NULL | Unique session name |
| SOURCE_DATABASE | VARCHAR2(128) | | Global name of the source database whose archived logs are to be mined in this persistent LogMiner session |
| SOURCE_DBID | NUMBER | | Database ID of the source database |
| SOURCE_RESETLOGS_SCN | NUMBER | | Resetlogs SCN associated with the incarnation of the source database whose archived logs are mined |
| SOURCE_RESETLOGS_TIME | NUMBER | | Resetlogs time associated with the incarnation of the source database whose archived logs are mined |
| FIRST_SCN | NUMBER | | Only modifications that occurred on or after this SCN can be mined using this persistent session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| END_SCN | NUMBER | | No modifications that occurred on or after this SCN can be mined using this persistent session |
| BRANCH_SCN | NUMBER | | SCN at which a branch will be taken in terms of the incarnation corresponding to the source database. This implies a point-in-time recovery was performed at the source database at this SCN. |
| WAIT_FOR_LOG | VARCHAR2(3) | | Indicates whether the persistent session waits for RFS to register new archived logs or to fill gaps (YES) or not (NO) |
| HOT_MINE | VARCHAR2(3) | | Indicates whether real-time mining is on (YES) or not (NO) |
| SAFE_PURGE_SCN | NUMBER | | Persistent session can safely be purged up to this SCN |
| CHECKPOINT_SCN | NUMBER | | SCN at which the latest checkpoint is taken by the persistent LogMiner session |

# DBA_LOGSTDBY_EDS_SUPPORTED

DBA_LOGSTDBY_EDS_SUPPORTED lists the tables that are candidates for EDS-based replication for Logical Standby based on the data types they contain.

In a CDB, the data displayed pertains to the container in which the view is queried.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(30) | | Table owner |
| TABLE_NAME | VARCHAR2(30) | | Table name |

# DBA_LOGSTDBY_EDS_TABLES

DBA_LOGSTDBY_EDS_TABLES lists the tables that have EDS-based replication for Logical Standby.

In a CDB, the data displayed pertains to the container in which the view is queried.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Schema name of supportable table |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Table name of supportable table |
| CTIME | TIMESTAMP(6) | | Time that the table had EDS added |

# DBA_LOGSTDBY_EVENTS

DBA_LOGSTDBY_EVENTS displays information about the activity of the logical standby database system. It can be used to determine the cause of failures that occur when applying redo data to logical standby databases. This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVENT_TIME | DATE | | Time when the event was logged |
| EVENT_TIMESTAMP | TIMESTAMP(6) | NOT NULL | Timestamp when the event was logged |
| START_SCN | NUMBER | | The SCN at which the associated transaction started at the primary database. This SCN refers to the system change number at the primary database. |
| CURRENT_SCN | NUMBER | | SCN associated with the change at the primary database. If a failure occurred, then examine this column to determine which archived log file contains the source of the failure (for example, an unsupported record). |
| COMMIT_SCN | NUMBER | | SCN value on which the change was committed at the primary database |
| XIDUSN | NUMBER | | Transaction ID undo segment number at the primary database of the associated transaction |
| XIDSLT | NUMBER | | Transaction ID slot number at the primary database of the associated transaction |
| XIDSQN | NUMBER | | Transaction ID sequence number at the primary database of the associated transaction |
| EVENT | CLOB | | Statement that was being processed when the failure occurred |
| STATUS_CODE | NUMBER | | Status (or Oracle error code) belonging to the STATUS message |
| STATUS | VARCHAR2(2000) | | Description of the current activity of the process or the reason why the apply operation stopped |
| SRC_CON_NAME | VARCHAR2(30) | | Identifies the PDB name at the primary database where the transaction was executed |
| SRC_CON_ID[1] | NUMBER | | Contains the PDB ID (the PDB_ID column from the DBA_PDBS view) of the primary database where the associated change was generated. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

In a CDB, this view shows data only when queried in the root.

# DBA_LOGSTDBY_HISTORY

DBA_LOGSTDBY_HISTORY displays the history of switchovers and failovers in a Data Guard configuration. It does this by showing the complete sequence of redo log streams processed or created on the local system, across all role transitions. (After a role transition, a new log stream is started and the log stream sequence number is

incremented by the new primary database.). This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STREAM_SEQUENCE# | NUMBER | | Lists the sequence numbers for all log streams created or applied on the local system.<br>**Note**: A value of 0 indicates an unknown sequence order; this is reserved for future log streams. |
| STATUS | VARCHAR2(16) | | Description of the log stream processing:<br>• `Past` - The log stream has already been processed<br>• `Immediate Past` - This is the most recently processed log stream; its status is transitioning from `Current` to `Past`<br>• `Current` - The log stream is currently being processed<br>• `Immediate Future` - This is the next log stream to be processed; its status is transitioning from `Future` to `Current`<br>• `Future` - The log stream will be processed |
| SOURCE | VARCHAR2(5) | | Describes how the log stream was started:<br>• `RFS` - The RFS process created the log stream<br>• `User` - A user registered the initial log file for the log stream<br>• `Synch` - A user issued the `ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY DDL` statement<br>• `Redo` - The log stream information was recorded in the redo log |
| DBID | NUMBER | | Database identifier of the primary database that created the log stream |
| FIRST_CHANGE# | NUMBER | | Lowest system change number (SCN) in the current log file |
| LAST_CHANGE# | NUMBER | | Highest system change number (SCN) in the current log file |
| FIRST_TIME | DATE | | Time of the first SCN entry (`FIRST_CHANGE#`) in the current log file |
| LAST_TIME | DATE | | Time of the last SCN entry (`LAST_CHANGE#`) in the current log file |
| DGNAME | VARCHAR2(255) | | Unique database name (`DB_UNIQUE_NAME`) of the primary database that produced the log stream. See `V$DATAGUARD_CONFIG` to display all database `DB_UNIQUE_NAME` values defined in the Data Guard configuration. |
| MERGE_CHANGE# | NUMBER | | SCN that should be used to flashback a failed primary (that created the log stream) or to flashback a bystander logical standby database following a failover, in the context of the associated redo log stream. It is the SCN up to |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | which redo for the associated log stream can be merged safely in all databases using local copies of archived logs received from the primary database. In order to apply changes beyond this following a failover, you will need to fetch and mine the redo logs from the failover target. |
| PROCESSED_CHANGE# | NUMBER | | Strict upper bound on the SCN up to which SQL Apply has applied redo records before it switched to a new log stream (either because it was activated and became the primary database, or in the case of a bystander logical standby database where it switched to a new log stream to accommodate a new primary database). |

**Note:**

In a CDB, this view shows data only when queried in the root.

# DBA_LOGSTDBY_LOG

DBA_LOGSTDBY_LOG displays information about the logs registered for a logical standby database. This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| THREAD# | NUMBER | NOT NULL | Thread ID of the archive log. The THREAD number is 1 for a single instance. For Real Application Clusters, this column will contain different numbers. |
| RESETLOGS_CHANGE# | NUMBER | NOT NULL | Start SCN of the branch |
| RESETLOGS_ID | NUMBER | NOT NULL | Resetlogs identifier (a numeric form of the timestamp of the branch) |
| SEQUENCE# | NUMBER | NOT NULL | Sequence number of the archive log file |
| FIRST_CHANGE# | NUMBER | NOT NULL | System change number (SCN) of the current archive log |
| NEXT_CHANGE# | NUMBER | | SCN of the next archive log |
| FIRST_TIME | DATE | | Date of the current archive log |
| NEXT_TIME | DATE | | Date of the next archive log |
| FILE_NAME | VARCHAR2(513) | | Name of the archive log |
| TIMESTAMP | DATE | | Time when the archive log was registered |
| DICT_BEGIN | VARCHAR2(3) | | Indicates whether the beginning of the dictionary build is in this archive log (YES) or not (NO) |
| DICT_END | VARCHAR2(3) | | Indicates whether the end of the dictionary build is in this archive log (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| APPLIED | VARCHAR2(8) | | Indicates primarily whether a given foreign archived log has been applied fully by SQL Apply:<br>• YES - SQL Apply has fully applied the foreign archived log and no longer needs it<br>• CURRENT - SQL Apply is currently applying changes contained in the foreign archived log<br>• NO - SQL Apply has not started applying any changes contained in the foreign archived log<br>• FETCHING - SQL Apply encountered a corruption while reading redo records from this foreign archived log, and is currently using the automatic gap resolution to refetch a new copy of the log from the primary database<br>• CORRUPT - SQL Apply encountered a corruption while reading redo records from this foreign archived log, and refetching a new copy of the archived log did not resolve the problem. SQL Apply will not refetch a new copy of this archived log automatically, and will require user intervention to manually register a new copy of the foreign archived log. |
| BLOCKS | NUMBER | | Number of blocks in the log |
| BLOCK_SIZE | NUMBER | | Size of each block in the log |

**Note:**

The SCN values in this view correlate to the SCN values shown in the DBA_LOGSTDBY_PROGRESS view.

**Note:**

In a CDB, this view shows data only when queried in the root.

# DBA_LOGSTDBY_NOT_UNIQUE

DBA_LOGSTDBY_NOT_UNIQUE displays all tables that have no primary and no non-null unique indexes. Most of the tables displayed by this view are supported because their columns contain enough information to be maintained in a logical standby database. Some tables, however, cannot be supported because their columns do not contain the necessary information. Unsupported tables usually contain a column defined using an unsupported data type.

In a CDB, the data displayed pertains to the container in which the view is queried.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(30) | | Schema name of the non-unique table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(30) | | Table name of the non-unique table |
| BAD_COLUMN | VARCHAR2(1) | | • Y - Table column is defined using an unbounded data type, such as LONG or BLOB. If two rows in the table match except in their LOB columns, then the table cannot be maintained properly. Log apply services will attempt to maintain these tables, but you must ensure the application does not allow uniqueness only in the unbounded columns.<br>• N - Enough column information is present to maintain the table in the logical standby database but the log transport services and log apply services would run more efficiently if you added a primary key. You should consider adding a disabled RELY constraint to these tables. |

# DBA_LOGSTDBY_PARAMETERS

DBA_LOGSTDBY_PARAMETERS displays the list of parameters used by SQL apply for logical standby databases. This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(64) | | Name of the parameter:<br>• MAX_SGA - System global area (SGA) allocated for the log apply services cache (in megabytes)<br>• MAX_SERVERS - Number of processes used by SQL Apply services<br>• PREPARE_SERVERS - Controls the number of parallel execution servers used to prepare changes<br>• APPLY_SERVERS - Controls the number of parallel execution servers used to apply changes<br>• MAX_EVENTS_RECORDED - Number of events stored in the DBA_LOGSTDBY_EVENTS view<br>• RECORD_SKIP_ERRORS - Indicates records that are skipped<br>• RECORD_SKIP_DDL - Indicates skipped DDL statements<br>• RECORD_APPLIED_DDL - Indicates applied DDL statements<br>• RECORD_UNSUPPORTED_OPERATIONS - Shows whether SQL Apply will capture information about transactions that did unsupported operations at the primary database in the DBA_LOGSTDBY_EVENTS view<br>• EVENT_LOG_DEST - Indicates where SQL Apply records the occurrence of an interesting event<br>• LOG_AUTO_DELETE - Shows whether SQL Apply will automatically delete remote archived logs received from the primary |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | database, once the contents of the logs are applied at the logical standby database.<br>• LOG_AUTO_DEL_RETENTION_TARGET - How many minutes a remote archived log received from the primary database will be retained at the logical standby database, once the contents of the log are applied by SQL Apply.<br>• PRESERVE_COMMIT_ORDER - Shows whether transactions are committed at the logical standby database in the same order that they were committed at the primary database |
| VALUE | VARCHAR2(2000) | | Value of the parameter |
| UNIT | VARCHAR2(64) | | Unit of the value, if applicable |
| SETTING | VARCHAR2(64) | | Possible values are as follows:<br>• SYSTEM - Parameter value was not explicitly set by the user. However, the user can change it with an appropriate call to the APPLY_SET procedure.<br>• USER - Parameter value was explicitly set by the user |
| DYNAMIC | VARCHAR2(64) | | YES if the parameter can be set dynamically (that is, without having to stop SQL Apply)<br>NO if setting the parameter requires that SQL Apply be stopped |

**Note:**

In a CDB, this view shows data when queried in the root.

# DBA_LOGSTDBY_PLSQL_MAP

DBA_LOGSTDBY_PLSQL_MAP shows the mapping between a supported user invokable (/external) PL/SQL procedure to the corresponding replicated internal PL/SQL procedure.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner name of the procedure |
| PKG_NAME | VARCHAR2(128) | | Package name of the user invokable procedure |
| PROC_NAME | VARCHAR2(128) | | Procedure name of the user invokable procedure |
| INTERNAL_PKG_NAME | VARCHAR2(128) | | Package name of the internal procedure |
| INTERNAL_PROC_NAME | VARCHAR2(128) | | Procedure name of the internal procedure |

**Note:**

In a CDB, this view shows data when queried in the root or a PDB.

# DBA_LOGSTDBY_PLSQL_SUPPORT

DBA_LOGSTDBY_PLSQL_SUPPORT shows the PL/SQL packages that are only supported during rolling operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner name of the package |
| PKG_NAME | VARCHAR2(128) | | Package name of the user invokable procedure |
| SUPPORT_LEVEL | VARCHAR2(12) | | Logical standby PL/SQL support level for the package:<br>• ALWAYS - PL/SQL replication is always supported for this package, whether it is called inside or outside of DBMS_ROLLING<br>• DBMS_ROLLING: PL/SQL replication is supported only when the procedure is called inside DBMS_ROLLING |

**Note:**

In a CDB, this view shows data when queried in the root or a PDB.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ROLLING package

# DBA_LOGSTDBY_PROGRESS

DBA_LOGSTDBY_PROGRESS is deprecated. The information that was provided in this view is now provided in the V$LOGSTDBY_PROGRESS view.

**See Also:**

"V$LOGSTDBY_PROGRESS"

# DBA_LOGSTDBY_SKIP

DBA_LOGSTDBY_SKIP displays the skip rules that are used by SQL Apply. This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ERROR | VARCHAR2(1) | | Indicates how the skip rule was created:<br>• Y - For rules from DBMS_LOGSTDBY.SKIP_ERROR |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • N - For rules from DBMS_LOGSTDBY.SKIP |
| STATEMENT_OPT | VARCHAR2(128) | | Specifies the type of statement that should be skipped |
| OWNER | VARCHAR2(128) | | Name of the schema under which the skip option should be used |
| NAME | VARCHAR2(261) | | Name of the object that is being skipped |
| USE_LIKE | VARCHAR2(1) | | Indicates whether the statement should use a SQL wildcard search when matching names (Y) or not (N) |
| ESC | VARCHAR2(1) | | Escape character used when performing wildcard matches |
| PROC | VARCHAR2(392) | | Name of a stored procedure that will be executed when processing the skip option |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_LOGSTDBY.SKIP_ERROR procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_LOGSTDBY.SKIP procedure

## DBA_LOGSTDBY_SKIP_TRANSACTION

DBA_LOGSTDBY_SKIP_TRANSACTION displays the skip settings chosen. This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| XIDUSN | NUMBER | | Transaction ID undo segment number |
| XIDSLT | NUMBER | | Transaction ID slot number |
| XIDSQN | NUMBER | | Transaction ID sequence number |

## DBA_LOGSTDBY_UNSUPPORTED

DBA_LOGSTDBY_UNSUPPORTED displays the schemas, tables, and columns in those tables that contain unsupported data types. The data displayed pertains to the container in which the view is queried. Use this view when you are preparing to perform a manual rolling upgrade using transient logical standby databases. This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(30) | | Owner of the unsupported table |
| TABLE_NAME | VARCHAR2(30) | | Name of the unsupported table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COLUMN_NAME | VARCHAR2(30) | | Name of the unsupported column |
| ATTRIBUTES | VARCHAR2(39) | | Displays the reason why the table is not supported by SQL Apply. If the structure of the table is unsupported (not just the datatype of some columns), then this view displays all columns in the table, with one of the following values displayed for each column in the table:<br>• IOT with Overflow<br>• IOT with LOB<br>• Mapping table for physical rowid of IOT<br>• IOT with row movement<br>• Table Compression<br>• Object Table<br>• AQ queue table<br>• Unsupported Virtual Column<br>• Encrypted Column<br>If the structure of the table is supported but certain columns in the table have unsupported datatypes, the ATTRIBUTE column will be NULL. |
| DATA_TYPE | VARCHAR2(32) | | Datatype of the unsupported column |

**Note:**

A rolling upgrade performed using the DBMS_ROLLING PL/SQL package supports more object types than a manual rolling upgrade performed using transient logical standby databases.

**See Also:**

- "DBA_ROLLING_UNSUPPORTED" for more information about determining unsupported data types for a rolling upgrade using the DBMS_ROLLING package
- *Oracle Data Guard Concepts and Administration* for more information about rolling operations
- *Oracle Data Guard Concepts and Administration* for more information about unsupported tables for rolling upgrade operations
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ROLLING package

# DBA_LOGSTDBY_UNSUPPORTED_TABLE

DBA_LOGSTDBY_UNSUPPORTED_TABLE displays the data tables that are not supported by Logical Standby. The data displayed pertains to the container in which the view is queried. This view is for logical standby databases only.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(30) | | Owner of the unsupported table |
| TABLE_NAME | VARCHAR2(30) | | Name of the unsupported table |

## DBA_MEASURE_FOLDER_CONTENTS

DBA_MEASURE_FOLDER_CONTENTS describes the contents of all OLAP measure folders in the database. Its columns are the same as those in ALL_MEASURE_FOLDER_CONTENTS.

**See Also:**

"ALL_MEASURE_FOLDER_CONTENTS"

## DBA_MEASURE_FOLDER_SUBFOLDERS

DBA_MEASURE_FOLDER_SUBFOLDERS describes the OLAP measure folders contained within the database OLAP measure folders. Its columns are the same as those in ALL_MEASURE_FOLDER_SUBFOLDERS.

**See Also:**

"ALL_MEASURE_FOLDER_SUBFOLDERS"

## DBA_MEASURE_FOLDERS

DBA_MEASURE_FOLDERS describes all OLAP measure folders in the database. Its columns are the same as those in ALL_MEASURE_FOLDERS.

**See Also:**

"ALL_MEASURE_FOLDERS"

## DBA_METADATA_PROPERTIES

DBA_METADATA_PROPERTIES describes OLAP metadata properties in the database. Its columns are the same as those in ALL_METADATA_PROPERTIES.

**See Also:**

"ALL_METADATA_PROPERTIES"

## DBA_METHOD_PARAMS

DBA_METHOD_PARAMS describes the method parameters of all object types in the database. Its columns are the same as those in ALL_METHOD_PARAMS.

**See Also:**

"ALL_METHOD_PARAMS"

## DBA_METHOD_RESULTS

DBA_METHOD_RESULTS describes the method results of all object types in the database. Its columns are the same as those in ALL_METHOD_RESULTS.

**See Also:**

"ALL_METHOD_RESULTS"

## DBA_MINING_MODEL_ATTRIBUTES

DBA_MINING_MODEL_ATTRIBUTES describes all mining model attributes in the database. Its columns are the same as those in ALL_MINING_MODEL_ATTRIBUTES.

**See Also:**

"ALL_MINING_MODEL_ATTRIBUTES"

## DBA_MINING_MODEL_SETTINGS

DBA_MINING_MODEL_SETTINGS describes all mining model settings in the database. Its columns are the same as those in ALL_MINING_MODEL_SETTINGS.

**See Also:**

"ALL_MINING_MODEL_SETTINGS"

## DBA_MINING_MODEL_TABLES

DBA_MINING_MODEL_TABLES describes the tables that contain metadata about the mining models in the database. Mining models are schema objects created by Oracle Data Mining.

Model tables reside in the schema of the mining model owner. The metadata stored in the tables is controlled by Oracle Data Mining APIs. The tables are read-only. They should not be modified by users.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the mining model |
| MODEL_NAME | VARCHAR2(128) | NOT NULL | Name of the mining model |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| TABLE_TYPE | VARCHAR2(21) | | The type of metadata stored in the table |

## DBA_MINING_MODELS

DBA_MINING_MODELS describes all mining models in the database. Its columns are the same as those in ALL_MINING_MODELS.

**See Also:**

"ALL_MINING_MODELS"

## DBA_MVIEW_AGGREGATES

DBA_MVIEW_AGGREGATES describes the grouping functions (aggregated measures) that appear in the SELECT list of all aggregated materialized view in the database. Its columns are the same as those in "ALL_MVIEW_AGGREGATES".

## DBA_MVIEW_ANALYSIS

DBA_MVIEW_ANALYSIS describes all materialized views in the database that potentially support query rewrite and that provide additional information for analysis by applications. Its columns are the same as those in "ALL_MVIEW_ANALYSIS".

**Note:**

This view excludes materialized views that reference remote tables or that include references to non-static values such as SYSDATE or USER. This view also excludes materialized views that were created as snapshots before Oracle8*i* and that were never altered to enable query rewrite.

## DBA_MVIEW_COMMENTS

DBA_MVIEW_COMMENTS displays comments on all materialized views in the database. Its columns are the same as those in ALL_MVIEW_COMMENTS.

**See Also:**

"ALL_MVIEW_COMMENTS"

## DBA_MVIEW_DETAIL_PARTITION

DBA_MVIEW_DETAIL_PARTITION displays freshness information for all materialized views in the database, with respect to a PCT detail partition. Its columns are the same as those in "ALL_MVIEW_DETAIL_PARTITION".

## DBA_MVIEW_DETAIL_RELATIONS

DBA_MVIEW_DETAIL_RELATIONS represents the named detail relations that are either in the FROM list of a materialized view, or that are indirectly referenced through views in the FROM list. Its columns are the same as those in "ALL_MVIEW_DETAIL_RELATIONS".

## DBA_MVIEW_DETAIL_SUBPARTITION

DBA_MVIEW_DETAIL_SUBPARTITION displays freshness information for all materialized views in the database, with respect to a PCT detail subpartition. Its columns are the same as those in "ALL_MVIEW_DETAIL_SUBPARTITION".

## DBA_MVIEW_JOINS

DBA_MVIEW_JOINS describes a join between two columns in the WHERE clause of a subquery that defines a materialized view. Its columns are the same as those in "ALL_MVIEW_JOINS".

## DBA_MVIEW_KEYS

DBA_MVIEW_KEYS describes the columns or expressions in the SELECT list upon which materialized views in the database are based. Its columns are the same as those in "ALL_MVIEW_KEYS".

## DBA_MVIEW_LOG_FILTER_COLS

DBA_MVIEW_LOG_FILTER_COLS displays all columns (excluding primary key columns) being logged in the materialized view logs.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the master table being logged |
| NAME | VARCHAR2(128) | | Name of the master table being logged |
| COLUMN_NAME | VARCHAR2(128) | | Column being logged |

## DBA_MVIEW_LOGS

DBA_MVIEW_LOGS describes all materialized view logs in the database. Its columns are the same as those in ALL_MVIEW_LOGS.

**See Also:**

"ALL_MVIEW_LOGS"

## DBA_MVIEW_REFRESH_TIMES

DBA_MVIEW_REFRESH_TIMES describes refresh times of all materialized views in the database. Its columns are the same as those in ALL_MVIEW_REFRESH_TIMES.

**See Also:**

"ALL_MVIEW_REFRESH_TIMES"

# DBA_MVIEWS

DBA_MVIEWS describes all materialized views in the database. Its columns are the same as those in ALL_MVIEWS.

**See Also:**

"ALL_MVIEWS"

# DBA_NESTED_TABLE_COLS

DBA_NESTED_TABLE_COLS describes the columns of all nested tables in the database. Its columns are the same as those in "ALL_NESTED_TABLE_COLS". To gather statistics for this view, use the DBMS_STATS package.

# DBA_NESTED_TABLES

DBA_NESTED_TABLES describes all nested tables contained in all tables in the database. Its columns are the same as those in "ALL_NESTED_TABLES".

# DBA_NETWORK_ACL_PRIVILEGES

DBA_NETWORK_ACL_PRIVILEGES describes the network privileges defined in all access control lists that are currently assigned to network hosts.

**Note:**

This DBA_NETWORK_ACL_PRIVILEGES view is deprecated in Oracle Database 12*c* Release 1 (12.1). Oracle recommends that you use the DBA_HOST_ACES view, instead.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ACL | VARCHAR2(4000) | | Path of the access control list |
| ACLID | RAW(8) | NOT NULL | Object ID of the access control list |
| PRINCIPAL | VARCHAR2(128) | | Principal (database user or role) whom the privilege is granted to or denied from |
| PRIVILEGE | VARCHAR2(128) | | Network privilege |
| IS_GRANT | VARCHAR2(5) | | Indicates whether the privilege is granted (true) or denied (false) |
| INVERT | VARCHAR2(5) | | Indicates whether the access control entry contains invert principal (true) or not (false) |
| START_DATE | TIMESTAMP(6) | | Start date of the access control entry |
| END_DATE | TIMESTAMP(6) | | End date of the access control entry |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ACL_OWNER | VARCHAR2(128) | | Owner of the access control list |

**See Also:**

"DBA_HOST_ACES"

# DBA_NETWORK_ACLS

DBA_NETWORK_ACLS describes the access control list assignments to network hosts.

**Note:**

This DBA_NETWORK_ACLS view is deprecated in Oracle Database 12*c* Release 1 (12.1). Oracle recommends that you use the DBA_HOST_ACLS view, instead.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HOST | VARCHAR2(1000) | NOT NULL | Network host |
| LOWER_PORT | NUMBER(5) | | Lower bound of the port range |
| UPPER_PORT | NUMBER(5) | | Upper bound of the port range |
| ACL | VARCHAR2(4000) | | Path of the access control list |
| ACLID | RAW(8) | | Object ID of the access control list |
| ACL_OWNER | VARCHAR2(128) | | Owner of the access control list |

**See Also:**

"DBA_HOST_ACLS"

# DBA_OBJ_AUDIT_OPTS

DBA_OBJ_AUDIT_OPTS describes auditing options on all objects.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

### Related View

`USER_OBJ_AUDIT_OPTS` describes auditing options on all objects owned by the current user. This view does not display the `OWNER` column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `OWNER` | `VARCHAR2(128)` | | Owner of the object |
| `OBJECT_NAME` | `VARCHAR2(128)` | | Name of the object |
| `OBJECT_TYPE` | `VARCHAR2(23)` | | Type of the object |
| `ALT` | `VARCHAR2(3)` | | Auditing `ALTER WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `AUD` | `VARCHAR2(3)` | | Auditing `AUDIT WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `COM` | `VARCHAR2(3)` | | Auditing `COMMENT WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `DEL` | `VARCHAR2(3)` | | Auditing `DELETE WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `GRA` | `VARCHAR2(3)` | | Auditing `GRANT WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `IND` | `VARCHAR2(3)` | | Auditing `INDEX WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `INS` | `VARCHAR2(3)` | | Auditing `INSERT WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `LOC` | `VARCHAR2(3)` | | Auditing `LOCK WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `REN` | `VARCHAR2(3)` | | Auditing `RENAME WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `SEL` | `VARCHAR2(3)` | | Auditing `SELECT WHENEVER SUCCESSFUL / UNSUCCESSFUL` |
| `UPD` | `VARCHAR2(3)` | | Auditing `UPDATE WHENEVER SUCCESSFUL / UNSUCCESSFUL` |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REF | CHAR(3) | | This column is obsolete and maintained for backward compatibility. The value of this column is always -/- |
| EXE | VARCHAR2(3) | | Auditing EXECUTE WHENEVER SUCCESSFUL / UNSUCCESSFUL |
| CRE | VARCHAR2(3) | | Auditing CREATE WHENEVER SUCCESSFUL / UNSUCCESSFUL |
| REA | VARCHAR2(3) | | Auditing READ WHENEVER SUCCESSFUL / UNSUCCESSFUL |
| WRI | VARCHAR2(3) | | Auditing WRITE WHENEVER SUCCESSFUL / UNSUCCESSFUL |
| FBK | VARCHAR2(3) | | Auditing FLASHBACK WHENEVER SUCCESSFUL / UNSUCCESSFUL |

**See Also:**

- "USER_OBJ_AUDIT_OPTS"
- *Oracle Database SQL Language Reference* for more information about the SQL AUDIT statement for unified auditing
- *Oracle Database SQL Language Reference* for more information about the SQL AUDIT statement for traditional auditing
- *Oracle Database Security Guide* to learn how to find information about audited activities

# DBA_OBJ_COLATTRS

DBA_OBJ_COLATTRS describes object columns and attributes contained in all tables in the database. Its columns are the same as those in ALL_OBJ_COLATTRS.

**See Also:**

"ALL_OBJ_COLATTRS"

# DBA_OBJECT_SIZE

DBA_OBJECT_SIZE lists the sizes, in bytes, of various PL/SQL objects.

### Related View

- USER_OBJECT_SIZE lists the size of PL/SQL objects owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| TYPE | VARCHAR2(18) | | Type of the object: TYPE, TYPE BODY, TABLE, VIEW, SYNONYM, SEQUENCE, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, JAVA SOURCE, JAVA CLASS or JAVA RESOURCE |
| SOURCE_SIZE | NUMBER | | Size of the source in bytes. Must be in memory during compilation, or dynamic recompilation. |
| PARSED_SIZE | NUMBER | | Size of the parsed form of the object, in bytes. Must be in memory when an object is being compiled that references this object. |
| CODE_SIZE | NUMBER | | Code size, in bytes. Must be in memory when this object is executing. |
| ERROR_SIZE | NUMBER | | Size of error messages, in bytes. In memory during the compilation of the object when there are compilation errors. |

**See Also:**

"USER_OBJECT_SIZE"

# DBA_OBJECT_TABLES

DBA_OBJECT_TABLES describes all object tables in the database. Its columns are the same as those in ALL_OBJECT_TABLES.

**See Also:**

"ALL_OBJECT_TABLES"

# DBA_OBJECT_USAGE

DBA_OBJECT_USAGE displays statistics about index usage gathered from the database for all the indexes in the database. You can use this view to monitor index usage. All indexes that have been used at least once can be monitored and displayed in this view.

### Related View

- USER_OBJECT_USAGE displays statistics about index usage gathered from the database for the indexes owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Index owner |
| INDEX_NAME | VARCHAR2(128) | NOT NULL | Index name in sys.obj$.name |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Table name in sys.obj$.name |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MONITORING | VARCHAR2(3) | | Indicates whether the monitoring feature is turned on. Possible values:<br>• YES<br>• NO |
| USED | VARCHAR2(3) | | Indicates whether the index has been accessed. Possible values:<br>• YES<br>• NO |
| START_MONITORING | VARCHAR2(19) | | Start monitoring time in sys.object_stats.start_monitoring |
| END_MONITORING | VARCHAR2(19) | | End monitoring time in sys.object_stats.end_monitoring |

**See Also:**

"USER_OBJECT_USAGE"

## DBA_OBJECTS

DBA_OBJECTS describes all objects in the database. Its columns are the same as those in "ALL_OBJECTS".

## DBA_OBJECTS_AE

DBA_OBJECTS_AE describes all objects (across all editions) in the database. Its columns are the same as those in ALL_OBJECTS_AE.

**See Also:**

"ALL_OBJECTS_AE"

## DBA_OPANCILLARY

DBA_OPANCILLARY provides ancillary information for all operator bindings in the database. Its columns are the same as those in "ALL_OPANCILLARY".

## DBA_OPARGUMENTS

DBA_OPARGUMENTS provides argument information for all operator bindings in the database. Its columns are the same as those in "ALL_OPARGUMENTS".

## DBA_OPBINDINGS

DBA_OPBINDINGS describes the binding functions and methods on all operators in the database. Its columns are the same as those in ALL_OPBINDINGS.

**See Also:**

"ALL_OPBINDINGS"

## DBA_OPERATOR_COMMENTS

DBA_OPERATOR_COMMENTS displays comments for all user-defined operators in the database. Its columns are the same as those in ALL_OPERATOR_COMMENTS.

**See Also:**

"ALL_OPERATOR_COMMENTS"

## DBA_OPERATORS

DBA_OPERATORS describes all operators in the database. Its columns are the same as those in ALL_OPERATORS.

**See Also:**

"ALL_OPERATORS"

## DBA_OPTSTAT_OPERATION_TASKS

DBA_OPTSTAT_OPERATION_TASKS displays the history of tasks that are performed as part of statistics operations (recorded in DBA_OPTSTAT_OPERATIONS). Each task represents a target object to be processed in the corresponding parent operation.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OPID | NUMBER | | Internal identifier for the statistics operation that the task belongs to |
| TARGET | VARCHAR2(100) | | Name of the object that this task operates on |
| TARGET_OBJN | NUMBER | | Object number of the target object |
| TARGET_TYPE | VARCHAR2(40) | | Type of the target object. Possible values are:<br>• TABLE<br>• TABLE (GLOBAL STATS ONLY): Task is created to gather only global statistics of a partitioned table<br>• TABLE (COORDINATOR JOB): Coordinator task for a partitioned table when concurrency is on<br>• TABLE PARTITION<br>• TABLE SUBPARTITION<br>• INDEX<br>• INDEX PARTITION<br>• INDEX SUBPARTITION |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TARGET_SIZE | NUMBER | | Target size (in number of blocks) when the task started |
| START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Task start time |
| END_TIME | TIMESTAMP(6) WITH TIME ZONE | | Task end time |
| STATUS | VARCHAR2(49) | | Current task status. Possible values are:<br>• PENDING: Task is queued for processing<br>• IN PROGRESS: Task is currently running<br>• COMPLETED: Task has completed successfully<br>• FAILED: Task has failed<br>• SKIPPED: Task has been skipped, as it does not exist any more, or its stats are not stale (applies only to only automatic statistics gathering)<br>• TIMED OUT: Maintenance window was not enough to complete this task (applies only to automatic statistics gathering) |
| JOB_NAME | VARCHAR2(50) | | Name of the scheduler job that executes this task (for example, when concurrency is on) |
| ESTIMATED_COST | NUMBER | | Estimated cost of the task (measured as elapsed time in seconds). This column is populated only when concurrency is on. |
| BATCHING_COEFF | NUMBER | | For internal use only |
| ACTIONS | NUMBER | | For internal use only |
| PRIORITY | NUMBER | | Rank of the task among all target objects for the parent operation |
| FLAGS | NUMBER | | For internal use only |
| NOTES | VARCHAR2(4000) | | Notes about the underlying task, such as the failure message for tasks with status FAILED. |

**See Also:**

"DBA_OPTSTAT_OPERATIONS"

# DBA_OPTSTAT_OPERATIONS

DBA_OPTSTAT_OPERATIONS contains a history of statistics operations performed at the schema and database level using the DBMS_STATS package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ID | NUMBER | | Internal ID of the statistics operation |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OPERATION | VARCHAR2(64) | | Operation name |
| TARGET | VARCHAR2(64) | | Target on which the operation was performed |
| START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time at which the operation started |
| END_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time at which the operation ended |
| STATUS | VARCHAR2(49) | | Current operation status. Possible values are:<br>• IN PROGRESS: Operation is currently running<br>• COMPLETED: Operation has completed successfully<br>• FAILED: Operation has failed<br>• TIMED OUT: Maintenance window was not enough to complete this operation (applies only to automatic statistics gathering) |
| JOB_NAME | VARCHAR2(32) | | Name of the scheduler job that executes this operation (for example, a user scheduled statistics gathering job) |
| SESSION_ID | NUMBER | | ID of the session in which this operation is invoked |
| NOTES | VARCHAR2(4000) | | Notes about the operation, such as a failure message for operations with status FAILED |

**See Also:**

- "DBA_OPTSTAT_OPERATION_TASKS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_REPAIR.ADMIN_TABLES procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_REPAIR.DUMP_ORPHAN_KEYS procedure

# DBA_ORPHAN_KEY_TABLE

DBA_ORPHAN_KEY_TABLE reports key values from indexes where the underlying base table has block corruptions. To create the view, run the DBMS_REPAIR.ADMIN_TABLES procedure. To populate the orphan key table for an index, run the DBMS_REPAIR.DUMP_ORPHAN_KEYS procedure on the index. For each

key in the index that points to a corrupt data block, Oracle inserts a row into the orphan key table.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA_NAME | VARCHAR2(30) | NOT NULL | Schema name of the index |
| INDEX_NAME | VARCHAR2(30) | NOT NULL | Name of the index |
| IPART_NAME | VARCHAR2(30) | NULL | Name of the index partition or subpartition |
| INDEX_ID | NUMBER | NOT NULL | Dictionary object ID of the index |
| TABLE_NAME | VARCHAR2(30) | NOT NULL | Name of the base table of the index |
| PART_NAME | VARCHAR2(30) | NULL | Name of the base table partition or subpartition |
| TABLE_ID | NUMBER | NOT NULL | Dictionary object ID of the base table |
| KEYROWID | ROWID | NOT NULL | Physical rowid of the corrupt data row |
| KEY | ROWID | NOT NULL | Key values for the index entry |
| DUMP_TIMESTAMP | DATE | NOT NULL | Timestamp when the entry was made into the orphan key table |

# DBA_OUTLINE_HINTS

DBA_OUTLINE_HINTS describes the set of hints stored in all outlines in the database.

### Related View

USER_OUTLINE_HINTS describes the set of hints stored in the outlines owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | | Name of the outline |
| OWNER | VARCHAR2(128) | | Name of the user who created the outline |
| NODE | NUMBER | | ID of the query or subquery to which the hint applies. The top-level query is labeled 1. Subqueries are assigned sequentially numbered labels, starting with 2. |
| STAGE | NUMBER | | Outline hints can be applied at three different stages during the compilation process. This column indicates the stage at which this hint was applied. |
| JOIN_POS | NUMBER | | Position of the table in the join order. The value is 0 for all hints except access method hints, which identify a table to which the hint and the join position apply. |
| HINT | CLOB | | Text of the hint |

**See Also:**

"USER_OUTLINE_HINTS"

# DBA_OUTLINES

DBA_OUTLINES describes all stored outlines in the database.

### Related View

USER_OUTLINES describes the stored outlines owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | | User-specified or generated name of the stored outline. The name must be of a form that can be expressed in SQL. |
| OWNER | VARCHAR2(128) | | Name of the user who created the outline |
| CATEGORY | VARCHAR2(128) | | User-defined name of the category to which the outline belongs |
| USED | VARCHAR2(6) | | Indicates whether the outline has ever been used (USED) or not (UNUSED) |
| TIMESTAMP | DATE | | Timestamp of outline creation |
| VERSION | VARCHAR2(64) | | Oracle version that created the outline |
| SQL_TEXT | LONG | | SQL text of the query, including any hints that were a part of the original statement. If bind variables are included, the variable names are stored as SQL text, not the values that are assigned to the variables.<br><br>**Note:** This field may contain sensitive information about your database or application. Therefore, use discretion when granting SELECT or VIEW object privileges on these views. |
| SIGNATURE | RAW(16) | | Signature uniquely identifying the outline SQL text |
| COMPATIBLE | VARCHAR2(12) | | Indicates whether the outline hints were compatible across a migration (COMPATIBLE) or not (INCOMPATIBLE) |
| ENABLED | VARCHAR2(8) | | Indicates whether the outline is enabled (ENABLED) or disabled (DISABLED) |
| FORMAT | VARCHAR2(6) | | Hint format:<br>• NORMAL<br>• LOCAL |
| MIGRATED | VARCHAR2(12) | | Indicates whether the outline has been migrated to a SQL plan baseline (MIGRATED) or not (NOT-MIGRATED) |

**See Also:**

"USER_OUTLINES"

# DBA_OUTSTANDING_ALERTS

DBA_OUTSTANDING_ALERTS describes alerts which the server considers to be outstanding.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEQUENCE_ID | NUMBER | | Alert sequence number |
| REASON_ID | NUMBER | NOT NULL | ID of the alert reason |
| OWNER | VARCHAR2(128) | | Owner of the object on which the alert was issued |
| OBJECT_NAME | VARCHAR2(513) | | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | | Name of the subobject (for example: partition) |
| OBJECT_TYPE | VARCHAR2(64) | | Object type (for example: table, tablespace) |
| REASON | VARCHAR2(4000) | | Reason for the alert |
| TIME_SUGGESTED | TIMESTAMP(6) WITH TIME ZONE | | Time when the alert was last updated |
| CREATION_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time when the alert was first created |
| SUGGESTED_ACTION | VARCHAR2(4000) | | Advice of the recommended action |
| ADVISOR_NAME | VARCHAR2(128) | | Name of the advisor to be invoked for more information |
| METRIC_VALUE | NUMBER | | Value of the related metrics |
| MESSAGE_TYPE | VARCHAR2(12) | | Message type:<br>• Notification<br>• Warning |
| MESSAGE_GROUP | VARCHAR2(64) | | Name of the message group to which the alert belongs |
| MESSAGE_LEVEL | NUMBER | | Message severity level (1 to 32) |
| HOSTING_CLIENT_ID | VARCHAR2(64) | | ID of the client or security group to which the alert relates |
| MODULE_ID | VARCHAR2(64) | | ID of the module that originated the alert |
| PROCESS_ID | VARCHAR2(128) | | Process ID |
| HOST_ID | VARCHAR2(256) | | DNS host name of the originating host |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HOST_NW_ADDR | VARCHAR2(256) | | IP or other network address of the originating host |
| INSTANCE_NAME | VARCHAR2(16) | | Originating instance name |
| INSTANCE_NUMBER | NUMBER | | Originating instance number |
| USER_ID | VARCHAR2(128) | | User ID |
| EXECUTION_CONTEXT_ID | VARCHAR2(128) | | Execution Context ID |
| ERROR_INSTANCE_ID | VARCHAR2(142) | | ID of an error instance plus a sequence number |
| STATE_TRANSITION_NUMBER | NUMBER | | Sequence number of the state transition for the alert |
| PDB_NAME | VARCHAR2(128) | | PDB name |
| CON_ID | NUMBER | NOT NULL | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# DBA_PARALLEL_EXECUTE_CHUNKS

DBA_PARALLEL_EXECUTE_CHUNKS displays the chunks for all tasks in the database.

### Related View

USER_PARALLEL_EXECUTE_CHUNKS displays the chunks for tasks created by the current user. This view does not display the TASK_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CHUNK_ID | NUMBER | NOT NULL | Unique ID for the chunk |
| TASK_OWNER | VARCHAR2(128) | NOT NULL | Owner of the task |
| TASK_NAME | VARCHAR2(128) | NOT NULL | Name of the task |
| STATUS | VARCHAR2(20) | | Status of the chunk:<br>• UNASSIGNED<br>• ASSIGNED<br>• PROCESSED<br>• PROCESSED_WITH_ERROR |
| START_ROWID | ROWID | | Rowid for the first row in the chunk |
| END_ROWID | ROWID | | Rowid for the last row in the chunk |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| START_ID | NUMBER | | Number column value of the first row in the chunk |
| END_ID | NUMBER | | Number column value of the last row in the chunk |
| JOB_NAME | VARCHAR2(128) | | Name of the job which processed this chunk |
| START_TS | TIMESTAMP(6) | | Processing start time for the chunk |
| END_TS | TIMESTAMP(6) | | Processing end time for the chunk |
| ERROR_CODE | NUMBER | | Error code returned during the execution of the chunk if the STATUS column is PROCESSED_WITH_ERROR |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message returned during the execution of the chunk if the STATUS column is PROCESSED_WITH_ERROR |

**See Also:**

"USER_PARALLEL_EXECUTE_CHUNKS"

# DBA_PARALLEL_EXECUTE_TASKS

DBA_PARALLEL_EXECUTE_TASKS displays all tasks in the database.

### Related View

USER_PARALLEL_EXECUTE_TASKS displays the tasks created by the current user. This view does not display the TASK_OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_OWNER | VARCHAR2(128) | NOT NULL | Owner of the task |
| TASK_NAME | VARCHAR2(128) | NOT NULL | Name of the task |
| CHUNK_TYPE | VARCHAR2(12) | | Type of parallel update:<br>• UNDELARED<br>• ROWID_RANGE<br>• NUMBER_RANGE |
| STATUS | VARCHAR2(19) | | Status of the task:<br>• CREATED<br>• CHUNKING<br>• CHUNKING_FAILED<br>• CHUNKED<br>• PROCESSING<br>• FINISHED<br>• FINISHED_WITH_ERROR<br>• CRASHED |
| TABLE_OWNER | VARCHAR2(128) | | Owner of the table to be chunked |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | | Name of the table to be chunked |
| NUMBER_COLUMN | VARCHAR2(128) | | Name of the column holding IDs (only applicable to NUMBER_RANGE chunking type) |
| TASK_COMMENT | VARCHAR2(4000) | | Comment field |
| JOB_PREFIX | VARCHAR2(128) | | Prefix of the job name executing this task |
| SQL_STMT | CLOB | | Argument used in the previous DBMS_PARALLEL_EXECUTE.RUN_TASK |
| LANGUAGE_FLAG | NUMBER | | Argument used in the previous DBMS_PARALLEL_EXECUTE.RUN_TASK |
| EDITION | VARCHAR2(130) | | Argument used in the previous DBMS_PARALLEL_EXECUTE.RUN_TASK |
| APPLY_CROSSEDITION_TRIGGER | VARCHAR2(130) | | Argument used in the previous DBMS_PARALLEL_EXECUTE.RUN_TASK |
| FIRE_APPLY_TRIGGER | VARCHAR2(10) | | Argument used in the previous DBMS_PARALLEL_EXECUTE.RUN_TASK |
| PARALLEL_LEVEL | NUMBER | | Argument used in the previous DBMS_PARALLEL_EXECUTE.RUN_TASK |
| JOB_CLASS | VARCHAR2(128) | | Argument used in the previous DBMS_PARALLEL_EXECUTE.RUN_TASK |

**See Also:**

- "USER_PARALLEL_EXECUTE_TASKS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PARALLEL_EXECUTE.RUN_TASK procedure

# DBA_PART_COL_STATISTICS

DBA_PART_COL_STATISTICS displays column statistics and histogram information for all table partitions in the database. Its columns are the same as those in ALL_PART_COL_STATISTICS.

**See Also:**

"ALL_PART_COL_STATISTICS"

# DBA_PART_HISTOGRAMS

DBA_PART_HISTOGRAMS displays the histogram data (endpoints per histogram) for the histograms on all table partitions in the database. Its columns are the same as those in ALL_PART_HISTOGRAMS.

**See Also:**

"ALL_PART_HISTOGRAMS"

## DBA_PART_INDEXES

DBA_PART_INDEXES displays the object-level partitioning information for all partitioned indexes in the database. Its columns are the same as those in ALL_PART_INDEXES.

**See Also:**

"ALL_PART_INDEXES"

## DBA_PART_KEY_COLUMNS

DBA_PART_KEY_COLUMNS describes the partitioning key columns for all partitioned objects in the database. Its columns are the same as those in ALL_PART_KEY_COLUMNS.

**See Also:**

"ALL_PART_KEY_COLUMNS"

## DBA_PART_LOBS

DBA_PART_LOBS displays table-level information for all partitioned LOBs in the database, including default attributes for LOB data partitions. Its columns are the same as those in ALL_PART_LOBS.

**See Also:**

"ALL_PART_LOBS"

## DBA_PART_TABLES

DBA_PART_TABLES displays the object-level partitioning information for all partitioned tables in the database. Its columns are the same as those in ALL_PART_TABLES.

**See Also:**

"ALL_PART_TABLES"

# DBA_PARTIAL_DROP_TABS

DBA_PARTIAL_DROP_TABS describes all tables in the database that have partially completed DROP COLUMN operations. Its columns are the same as those in "ALL_PARTIAL_DROP_TABS".

# DBA_PDB_HISTORY

DBA_PDB_HISTORY describes the lineage of the PDB to which it belongs.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PDB_NAME | VARCHAR2(128) | NOT NULL | Name of this PDB in one of its incarnations |
| PDB_ID | NUMBER | NOT NULL | Container ID of this PDB in one of its incarnations. |
| PDB_DBID | NUMBER | NOT NULL | Database ID of this PDB in one of its incarnations |
| PDB_GUID | RAW(16) | NOT NULL | Globally unique ID of this PDB in one of its incarnations |
| OP_SCNBAS | NUMBER | NOT NULL | SCN base when an operation was performed on one of the incarnations of this PDB |
| OP_SCNWRP | NUMBER | NOT NULL | SCN wrap when an operation was performed on one of incarnations of this PDB |
| OP_TIMESTAMP | DATE | NOT NULL | Timestamp of an operation performed on one of the incarnations of this PDB |
| OPERATION | VARCHAR2(16) | NOT NULL | Operation that was performed on one of the incarnations of this PDB |
| DB_VERSION | NUMBER | NOT NULL | Database version |
| CLONED_FROM_PDB_NAME | VARCHAR2(128) | | Name of a PDB from which one of the incarnations of this PDB was cloned |
| CLONED_FROM_PDB_DBID | NUMBER | | Database ID of a PDB from which one of the incarnations of this PDB was cloned |
| CLONED_FROM_PDB_GUID | RAW(16) | | Globally unique ID of a PDB from which one of the incarnations of this PDB was cloned |
| DB_NAME | VARCHAR2(128) | | Name of a CDB in which one of the incarnations of this PDB was created |
| DB_UNIQUE_NAME | VARCHAR2(128) | | Unique name of a CDB in which one of the incarnations of this PDB was created |
| DB_DBID | NUMBER | | Database ID of a CDB in which one of the incarnations of this PDB was created |
| CLONETAG | VARCHAR2(128) | | Clone tag name for the PDB if the PDB was cloned using the snapshot copy mechanism |

# DBA_PDB_SAVED_STATES

DBA_PDB_SAVED_STATES shows information about the current saved PDB states in the CDB.

This view is an object link, so the data is also available within the PDB.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CON_ID | NUMBER | NOT NULL | The ID of the PDB |
| CON_NAME | VARCHAR2(128) | NOT NULL | Name of the PDB |
| INSTANCE_NAME | VARCHAR2(128) | NOT NULL | Name of the instance for which the state is saved |
| CON_UID | NUMBER | NOT NULL | Unique ID assigned to the PDB at creation time |
| GUID | RAW(16) | | Globally unique immutable ID assigned to the PDB at creation time |
| STATE | VARCHAR2(14) | | Open state of the PDB |
| RESTRICTED | VARCHAR2(3) | | Restricted mode of the PDB |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

*Oracle Database SQL Language Reference* for more information about preserving a PDB's open mode across an instance restart

# DBA_PDBS

DBA_PDBS describes PDBs belonging to a given CDB. When queried from the root, it will describe all PDBs belonging to a given CDB. When queried from a PDB, it will always return no rows.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PDB_ID | NUMBER | NOT NULL | Container ID of the PDB |
| PDB_NAME | VARCHAR2(128) | NOT NULL | Name of the PDB |
| DBID | NUMBER | NOT NULL | PDB identifier calculated when the PDB is created and stored in all file headers associated with the PDB |
| CON_UID | NUMBER | NOT NULL | Unique identifier associated with the container |
| GUID | RAW(16) | | Globally unique immutable ID assigned to the PDB at creation time |
| STATUS | VARCHAR2(13) | | State of the PDB. Possible values: |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NEW - The PDB has never been opened since it was created. It must be opened in READ WRITE mode for Oracle to perform processing needed to complete the integration of the PDB into the CDB and mark it NORMAL. An error will be thrown if an attempt is made to open the PDB read only.<br>• NORMAL - The PDB is ready to be used.<br>• UNPLUGGED - The PDB has been unplugged. The only operation that can be performed on it is DROP PLUGGABLE DATABASE.<br>• NEEDS UPGRADE - A PDB needs to be upgraded to the version of the CDB into which it was plugged.<br>• CONVERTING - A non-CDB was plugged into the CDB and is undergoing conversion required to make it behave like a real PDB.<br>• UNUSABLE - The PDB is being created or an unrecoverable error was encountered during its creation. The PDB cannot be opened while its state is set to UNUSABLE. If the PDB remains in this state because of an error encountered during its creation, it can only be dropped. The alert log can be checked to determine if there was an error during PDB creation. |
| CREATION_SCN | NUMBER | | Creation SCN |
| VSN | NUMBER | | The version number of the PDB |
| LOGGING[1] | VARCHAR2(9) | | Shows the current logging mode for the PDB. Possible values:<br>• LOGGING<br>• NOLOGGING |
| FORCE_LOGGING[1] | VARCHAR2(3) | | Specifies whether force logging is turned on for the PDB. Possible values:<br>• NO<br>• YES |
| FORCE_NOLOGGING[1] | VARCHAR2(3) | | Specifies whether force nologging is turned on for the PDB. Possible values:<br>• NO<br>• YES |
| CON_ID | NUMBER | NOT NULL | The ID of the container that CON_DBID identifies. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# DBA_PENDING_CONV_TABLES

DBA_PENDING_CONV_TABLES describes all pending conversion tables in the database. Its columns are the same as those in ALL_PENDING_CONV_TABLES.

**See Also:**

"ALL_PENDING_CONV_TABLES"

# DBA_PENDING_TRANSACTIONS

DBA_PENDING_TRANSACTIONS describes unresolved transactions (either due to failure or if the coordinator has not sent a commit/rollback).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FORMATID | NUMBER | | The format identifier of the transaction identifier |
| GLOBALID | RAW(64) | | The global part (gtrid) of the transaction identifier |
| BRANCHID | RAW(64) | | The branch qualifier (bqual) of the transaction identifier |

# DBA_PLSQL_COLL_TYPES

DBA_PLSQL_COLL_TYPES describes all named PL/SQL collection types in the database. Its columns (except for CHAR_USED) are the same as those in ALL_PLSQL_COLL_TYPES.

**See Also:**

"ALL_PLSQL_COLL_TYPES"

# DBA_PLSQL_OBJECT_SETTINGS

DBA_PLSQL_OBJECT_SETTINGS displays information about the compiler settings for all stored objects in the database. Its columns are the same as those in ALL_PLSQL_OBJECT_SETTINGS.

**See Also:**

"ALL_PLSQL_OBJECT_SETTINGS"

# DBA_PLSQL_TYPE_ATTRS

DBA_PLSQL_TYPE_ATTRS describes the attributes of all PL/SQL types in the database. Its columns are the same as those in ALL_PLSQL_TYPE_ATTRS.

**See Also:**

"ALL_PLSQL_TYPE_ATTRS"

## DBA_PLSQL_TYPES

DBA_PLSQL_TYPES describes all PL/SQL types in the database. Its columns are the same as those in ALL_PLSQL_TYPES.

**See Also:**

"ALL_PLSQL_TYPES"

## DBA_POLICIES

DBA_POLICIES describes all Oracle Virtual Private Database (VPD) security policies in the database. Its columns are the same as those in ALL_POLICIES. A security policy is a list of security requirements and rules that regulate row level access to database objects.

**See Also:**

"ALL_POLICIES"

## DBA_POLICY_ATTRIBUTES

DBA_POLICY_ATTRIBUTES lists the attribute associations {Namespaces, Attributes} of all context-sensitive and shared context-sensitive Oracle Virtual Private Database (VPD) policies in the database. Its columns are the same as those in ALL_POLICY_ATTRIBUTES.

**See Also:**

"ALL_POLICY_ATTRIBUTES"

## DBA_POLICY_CONTEXTS

DBA_POLICY_CONTEXTS describes all driving contexts in the database. Its columns are the same as those in ALL_POLICY_CONTEXTS.

**See Also:**

"ALL_POLICY_CONTEXTS"

## DBA_POLICY_GROUPS

DBA_POLICY_GROUPS describes all policy groups in the database. Its columns are the same as those in ALL_POLICY_GROUPS.

**See Also:**

"ALL_POLICY_GROUPS"

# DBA_PRIV_AUDIT_OPTS

DBA_PRIV_AUDIT_OPTS describes current system privileges being audited across the system and by user.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_NAME | VARCHAR2(128) | | User name if by user auditing; ANY CLIENT if access by a proxy on behalf of a client is being audited; NULL for systemwide auditing |
| PROXY_NAME | VARCHAR2(128) | | Name of the proxy user which is performing an operation for the client; NULL if the client is performing the operation directly |
| PRIVILEGE | VARCHAR2(40) | NOT NULL | Name of the system privilege being audited |
| SUCCESS | VARCHAR2(10) | | Mode for WHENEVER SUCCESSFUL system auditing |
| FAILURE | VARCHAR2(10) | | Mode for WHENEVER NOT SUCCESSFUL system auditing |

# DBA_PRIV_CAPTURES

DBA_PRIV_CAPTURES lists the privilege analysis policies in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | NOT NULL | Name of the privilege analysis policy |
| DESCRIPTION | VARCHAR2(1024) | | Description of the privilege analysis |
| TYPE | VARCHAR2(16) | | Type of the privilege analysis policy.<br>Possible values:<br>• G_DATABASE: Database wide privilege analysis |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • G_ROLE: Role privilege analysis<br>• G_CONTEXT: Context privilege analysis<br>• G_ROLE_AND_CONTEXT: Role and context privilege analysis |
| ENABLED | VARCHAR2(1) | | Enabling status of the privilege analysis |
| ROLES | ROLE_ID_LIST | | List of roles whose privileges to analyze if the privilege analysis type is G_ROLE or G_ROLE_AND_CONTEXT |
| CONTEXT | VARCHAR2(4000) | | Context condition if the privilege analysis type is G_CONTEXT or G_ROLE_AND_CONTEXT |

**See Also:**

*Oracle Database Vault Administrator's Guide* for more information about privilege analysis

# DBA_PROCEDURES

DBA_PROCEDURES lists all functions and procedures that are available in the database, along with their associated properties. Its columns are the same as those in ALL_PROCEDURES.

**See Also:**

- "ALL_PROCEDURES"
- "DBA_ARGUMENTS" for information about the arguments of all of the functions and procedures that are available in the database

# DBA_PROFILES

DBA_PROFILES displays all profiles and their limits.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROFILE | VARCHAR2(128) | NOT NULL | Profile name |
| RESOURCE_NAME | VARCHAR2(32) | NOT NULL | Resource name |
| RESOURCE_TYPE | VARCHAR2(8) | | Indicates whether the resource profile is a KERNEL or a PASSWORD parameter |
| LIMIT | VARCHAR2(128) | | Limit placed on this resource for this profile |
| COMMON | VARCHAR2(3) | | Indicates whether a given profile is common. Possible values:<br>• YES if a profile is common<br>• NO if a profile is local (not common) |

# DBA_PROPAGATION

DBA_PROPAGATION displays information about all Streams propagations in the database. Its columns are the same as those in ALL_PROPAGATION.

**See Also:**

"ALL_PROPAGATION"

# DBA_PROXIES

DBA_PROXIES displays Information about all proxy connections in the database.

### Related View

USER_PROXIES displays information about connections the current user is allowed to proxy. This view does not display the PROXY or PROXY_AUTHORITY columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROXY | VARCHAR2(128) | | Name of the proxy user |
| CLIENT | VARCHAR2(128) | NOT NULL | Name of the client user who the proxy user can act on behalf of |
| AUTHENTICATION | VARCHAR2(3) | | Indicates whether the proxy is required to supply the client's authentication credentials (YES) or not (NO) |
| AUTHORIZATION_CONSTRAINT | VARCHAR2(35) | | Indicates the proxy's authority to exercise roles on the client's behalf:<br>• NO CLIENT ROLES MAY BE ACTIVATED<br>• PROXY MAY ACTIVATE ROLE<br>• PROXY MAY ACTIVATE ALL CLIENT ROLES<br>• PROXY MAY NOT ACTIVATE ROLE |
| ROLE | VARCHAR2(128) | | Name of the role referenced in AUTHORIZATION_CONSTRAINT |
| PROXY_AUTHORITY | VARCHAR2(9) | | Value is either:<br>• DIRECTORY if EUS proxy is enabled for that database user<br>• DATABASE if this row describes a local proxy permission |

**See Also:**

"USER_PROXIES"

# DBA_QUEUE_SCHEDULES

DBA_QUEUE_SCHEDULES describes all propagation schedules in the database. Its columns are the same as those in ALL_QUEUE_SCHEDULES.

**See Also:**

"ALL_QUEUE_SCHEDULES"

## DBA_QUEUE_SUBSCRIBERS

DBA_QUEUE_SUBSCRIBERS displays all subscribers on all queues in the database. Its columns are the same as those in "ALL_QUEUE_SUBSCRIBERS".

## DBA_QUEUE_TABLES

DBA_QUEUE_TABLES contains information about the owner instance for a queue table. A queue table can contain multiple queues. In this case, each queue in a queue table has the same owner instance as the queue table. Its columns are the same as those in ALL_QUEUE_TABLES.

**See Also:**

"ALL_QUEUE_TABLES"

## DBA_QUEUES

DBA_QUEUES describes the operational characteristics of every queue in a database. Its columns are the same as those in "ALL_QUEUES".

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Advanced Queuing

## DBA_RCHILD

DBA_RCHILD displays all the children in any refresh group.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REFGROUP | NUMBER | | Internal identifier of the refresh group |
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object in the refresh group |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object in the refresh group |
| TYPE# | VARCHAR2(128) | | Type of the object in the refresh group |

## DBA_RECOVERABLE_SCRIPT

DBA_RECOVERABLE_SCRIPT provides details about recoverable operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCRIPT_ID | RAW(16) | | Unique ID of the operation |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CREATION_TIME | DATE | | Time the operation was invoked |
| INVOKING_PACKAGE_OWNER | VARCHAR2(128) | | Invoking package owner of the operation |
| INVOKING_PACKAGE | VARCHAR2(128) | | Invoking package of the operation |
| INVOKING_PROCEDURE | VARCHAR2(128) | | Invoking procedure of the operation |
| INVOKING_USER | VARCHAR2(128) | | Script owner |
| STATUS | VARCHAR2(12) | | State of the recoverable script: GENERATING, NOT EXECUTED, EXECUTING, EXECUTED, or ERROR |
| TOTAL_BLOCKS | NUMBER | | Total number of blocks for the recoverable script to be executed |
| DONE_BLOCK_NUM | NUMBER | | Last block executed, thus far |
| SCRIPT_COMMENT | VARCHAR2(4000) | | Comment for the recoverable script |

# DBA_RECOVERABLE_SCRIPT_BLOCKS

DBA_RECOVERABLE_SCRIPT_BLOCKS provides details about recoverable script blocks.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCRIPT_ID | RAW(16) | | Global unique ID of the recoverable script to which this block belongs |
| BLOCK_NUM | NUMBER | | The *n*th block in the recoverable script to be executed |
| FORWARD_BLOCK | CLOB | | Forward block to be executed |
| FORWARD_BLOCK_DBLINK | VARCHAR2(128) | | Database where the forward block is executed |
| UNDO_BLOCK | CLOB | | Block to roll back the forward operation |
| UNDO_BLOCK_DBLINK | VARCHAR2(128) | | Database where the undo block is executed |
| STATUS | VARCHAR2(12) | | Status of the block execution: GENERATING, NOT EXECUTED, EXECUTING, EXECUTED, or ERROR |
| BLOCK_COMMENT | VARCHAR2(4000) | | Comment for the block |

# DBA_RECOVERABLE_SCRIPT_ERRORS

DBA_RECOVERABLE_SCRIPT_ERRORS provides details about errors that occurred during script execution.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCRIPT_ID | RAW(16) | | Global unique ID of the recoverable script |
| BLOCK_NUM | NUMBER | | The *n*th block that failed |
| ERROR_NUMBER | NUMBER | | Number of the error encountered while executing the block |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message encountered while executing the block |
| ERROR_CREATION_TIME | DATE | | Time that the error was created |

# DBA_RECOVERABLE_SCRIPT_HIST

DBA_RECOVERABLE_SCRIPT_HIST displays details about executed or purged recoverable operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCRIPT_ID | RAW(16) | | Unique id of the operation |
| CREATION_TIME | DATE | | Time the operation was invoked |
| INVOKING_PACKAGE_OWNER | VARCHAR2(128) | | Invoking package owner of the operation |
| INVOKING_PACKAGE | VARCHAR2(128) | | Invoking package of the operation |
| INVOKING_PROCEDURE | VARCHAR2(128) | | Invoking procedure of the operation |
| INVOKING_USER | VARCHAR2(128) | | Script owner |
| STATUS | VARCHAR2(8) | | state of the recoverable script: EXECUTED, PURGED |
| TOTAL_BLOCKS | NUMBER | | total number of blocks for the recoverable script to be executed |
| DONE_BLOCK_NUM | NUMBER | | last block so far executed |
| SCRIPT_COMMENT | VARCHAR2(4000) | | comment for the recoverable script |

# DBA_RECOVERABLE_SCRIPT_PARAMS

DBA_RECOVERABLE_SCRIPT_PARAMS provides details about recoverable operation parameters.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCRIPT_ID | RAW(16) | | Unique ID of the operation |
| PARAMETER | VARCHAR2(128) | | Name of the parameter |
| PARAM_INDEX | NUMBER | | Index for multi-valued parameter |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| VALUE | VARCHAR2(4000) | | Value of the parameter |

# DBA_RECYCLEBIN

DBA_RECYCLEBIN displays information about all recycle bins in the database.

### Related View

USER_RECYCLEBIN displays information about the recycle bin owned by the current user. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Name of the original owner of the object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | New name of the object |
| ORIGINAL_NAME | VARCHAR2(128) | | Original name of the object |
| OPERATION | VARCHAR2(9) | | Operation carried out on the object:<br>• DROP - Object was dropped<br>• TRUNCATE - Object was truncated<br>**Note:** The Oracle Database currently only supports recovering dropped objects from the recycle bin. The truncated objects cannot be recovered. |
| TYPE | VARCHAR2(25) | | Type of the object:<br>• TABLE<br>• NORMAL INDEX<br>• BITMAP INDEX<br>• NESTED TABLE<br>• LOB<br>• LOB INDEX<br>• DOMAIN INDEX<br>• IOT TOP INDEX<br>• IOT OVERFLOW SEGMENT<br>• IOT MAPPING TABLE<br>• TRIGGER<br>• Table Partition<br>• Table Composite Partition<br>• Index Partition<br>• Index Composite Partition<br>• LOB Partition<br>• LOB Composite Partition |
| TS_NAME | VARCHAR2(30) | | Name of the tablespace to which the object belongs |
| CREATETIME | VARCHAR2(19) | | Timestamp for the creation of the object |
| DROPTIME | VARCHAR2(19) | | Timestamp for the dropping of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DROPSCN | NUMBER | | System change number (SCN) of the transaction which moved the object to the recycle bin |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition which was dropped |
| CAN_UNDROP | VARCHAR2(3) | | Indicates whether the object can be undropped (YES) or not (NO) |
| CAN_PURGE | VARCHAR2(3) | | Indicates whether the object can be purged (YES) or not (NO) |
| RELATED | NUMBER | NOT NULL | Object number of the parent object |
| BASE_OBJECT | NUMBER | NOT NULL | Object number of the base object |
| PURGE_OBJECT | NUMBER | NOT NULL | Object number for the object which gets purged |
| SPACE | NUMBER | | Number of blocks used by the object |

**See Also:**

"USER_RECYCLEBIN"

# DBA_REDEFINITION_ERRORS

DBA_REDEFINITION_ERRORS is an online redefinition view. It displays the dependent objects for which errors were raised while attempting to create similar objects on the interim table of the redefinition.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_TYPE | VARCHAR2(12) | | Type of the redefinition object:<br>• TABLE<br>• INDEX<br>• CONSTRAINT<br>• TRIGGER<br>• NESTED TABLE<br>• PARTITION<br>• MV LOG |
| OBJECT_OWNER | VARCHAR2(4000) | | Owner of the redefinition object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the redefinition object |
| BASE_TABLE_OWNER | VARCHAR2(128) | | Owner of the base table of the redefinition object |
| BASE_TABLE_NAME | VARCHAR2(128) | | Name of the base table of the redefinition object |
| DDL_TXT | CLOB | | DDL used to create the corresponding interim redefinition object |
| EDITION_NAME | VARCHAR2(128) | | Reserved for future use |
| ERR_NO | NUMBER(38) | | Oracle error number corresponding to this error |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ERR_TXT | VARCHAR2(1000) | | Oracle error text corresponding to this error |

**See Also:**

*Oracle Database Administrator's Guide* for more information about online redefinition

# DBA_REDEFINITION_OBJECTS

DBA_REDEFINITION_OBJECTS is an online redefinition view. It displays the objects involved in the current redefinitions.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_TYPE | VARCHAR2(12) | | Type of the redefinition object:<br>• TABLE<br>• INDEX<br>• CONSTRAINT<br>• TRIGGER<br>• NESTED TABLE<br>• PARTITION<br>• MV LOG |
| OBJECT_OWNER | VARCHAR2(4000) | | Owner of the redefinition object |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the redefinition object |
| BASE_TABLE_OWNER | VARCHAR2(128) | | Owner of the base table of the redefinition object |
| BASE_TABLE_NAME | VARCHAR2(128) | | Name of the base table of the redefinition object |
| INTERIM_OBJECT_OWNER | VARCHAR2(4000) | | Owner of the corresponding interim redefinition object |
| INTERIM_OBJECT_NAME | VARCHAR2(128) | | Name of the corresponding interim redefinition object |
| EDITION_NAME | VARCHAR2(128) | | Reserved for future use |

**See Also:**

*Oracle Database Administrator's Guide* for more information about online redefinition

# DBA_REDEFINITION_STATUS

DBA_REDEFINITION_STATUS is an online redefinition view. It provides information about the online redefinition status.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REDEFINITION_ID | NUMBER(38) | NOT NULL | ID for the redefinition object |
| BASE_TABLE_OWNER | VARCHAR2(128) | | Owner of the base table of the redefinition object |
| BASE_TABLE_NAME | VARCHAR2(128) | | Name of the base table of the redefinition object |
| BASE_PARTITION_NAME | VARCHAR2(128) | NOT NULL | Name of the base partition of the redefinition object |
| INTERIM_OBJECT_OWNER | VARCHAR2(128) | | Owner of the interim object of the redefinition object |
| INTERIM_OBJECT_NAME | VARCHAR2(128) | | Name of the interim object of the redefinition object |
| OPERATION[1] | VARCHAR2(128) | NOT NULL | The current redefinition operation:<br>• START_REDEF_TABLE<br>• SYNC_INTERIM_TABLE<br>• COPY_TABLE_DEPENDENTS<br>• FINISH_REDEF_TABLE |
| STATUS | VARCHAR2(128) | NOT NULL | Status of the previous redefinition operation:<br>• FAILURE<br>• SUCCESS |
| RESTARTABLE | VARCHAR2(1) | NOT NULL | Indicates whether the previous operation can be restarted |

1 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

*Oracle Database Administrator's Guide* for more information about online redefinition

# DBA_REFRESH

DBA_REFRESH describes all refresh groups in the database. Its columns are the same as those in "ALL_REFRESH".

# DBA_REFRESH_CHILDREN

DBA_REFRESH_CHILDREN lists all of the objects in all refresh groups in the database. Its columns are the same as those in "ALL_REFRESH_CHILDREN".

# DBA_REFS

DBA_REFS describes the REF columns and REF attributes in object type columns of all the objects in the database. Its columns are the same as those in "ALL_REFS".

# DBA_REGISTERED_ARCHIVED_LOG

DBA_REGISTERED_ARCHIVED_LOG displays information about all registered archived logfiles in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CONSUMER_NAME | VARCHAR2(128) | NOT NULL | Consumer name of the archived logs |
| SOURCE_DATABASE | VARCHAR2(128) | | Name of the database which generated the redo logs |
| THREAD# | NUMBER | NOT NULL | Thread number of the archived redo log. The thread number is 1 for a single instance. For Real Application Clusters, this column will contain different numbers. |
| SEQUENCE# | NUMBER | NOT NULL | Sequence number of the archived redo log file |
| FIRST_SCN | NUMBER | NOT NULL | System change number (SCN) of the current archived redo log |
| NEXT_SCN | NUMBER | | System change number (SCN) of the next archived redo log |
| FIRST_TIME | DATE | | Date and time of the current archived redo log |
| NEXT_TIME | DATE | | Date and time of the next archived redo log |
| NAME | VARCHAR2(513) | | Name of the archived redo log |
| MODIFIED_TIME | DATE | | Time when the archived redo log was registered |
| DICTIONARY_BEGIN | VARCHAR2(3) | | Indicates whether the beginning of the dictionary build is in the archived redo log (YES) or not (NO) |
| DICTIONARY_END | VARCHAR2(3) | | Indicates whether the end of the dictionary build is in the archived redo log (YES) or not (NO) |
| PURGEABLE | VARCHAR2(3) | | Indicates whether the redo log can be permanently removed (YES) or not (NO) |
| RESET_LOGS_CHANGE# | NUMBER | NOT NULL | Resetlogs change number of the database when the log was written |
| RESET_TIMESTAMP | NUMBER | NOT NULL | Resetlogs time of the database when the log was written |

# DBA_REGISTERED_MVIEW_GROUPS

DBA_REGISTERED_MVIEW_GROUPS lists all the registered materialized view groups at the master site or master materialized view site.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(30) | | Name of the materialized view replication group |
| MVIEW_SITE | VARCHAR2(128) | | Site of the materialized view replication group |
| GROUP_COMMENT | VARCHAR2(80) | | Description of the materialized view replication group |
| VERSION | VARCHAR2(8) | | Oracle release of the materialized view replication group:<br>• ORACLE 7<br>• ORACLE 8 |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • REPAPI<br>**Note:** Oracle8*i* and newer materialized view groups show ORACLE 8. |
| FNAME | VARCHAR2(30) | | Name of the flavor of the materialized view group |
| OWNER | VARCHAR2(128) | | Owner of the materialized view replication group |

# DBA_REGISTERED_MVIEWS

DBA_REGISTERED_MVIEWS describes all registered materialized views (registered at a master site or a master materialized view site) in the database. Its columns are the same as those in ALL_REGISTERED_MVIEWS.

**See Also:**

"ALL_REGISTERED_MVIEWS"

# DBA_REGISTRY

DBA_REGISTRY displays information about all components in the database that are loaded into the component registry. The component registry tracks components that can be separately loaded into the Oracle Database. When a SQL script loads the PL/SQL packages and other database objects for a component into the database, the script records the component name, status, and version. If scripts are used to upgrade/downgrade the dictionary elements for the component, then those scripts also record status and version information.

### Related View

USER_REGISTRY displays information about the components owned by the current user that are loaded into the component registry.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMP_ID | VARCHAR2(30) | NOT NULL | Component identifier |
| COMP_NAME | VARCHAR2(255) | | Component name |
| VERSION | VARCHAR2(30) | | Component version loaded |
| STATUS | VARCHAR2(11) | | Component status:<br>• INVALID<br>• VALID<br>• LOADING<br>• LOADED<br>• UPGRADING<br>• UPGRADED<br>• DOWNGRADING<br>• DOWNGRADED<br>• REMOVING<br>• REMOVED |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MODIFIED | VARCHAR2(20) | | Time when the component was last modified |
| NAMESPACE | VARCHAR2(30) | NOT NULL | Component namespace |
| CONTROL | VARCHAR2(128) | NOT NULL | User that created the component entry |
| SCHEMA | VARCHAR2(128) | NOT NULL | User that contains the objects for the component |
| PROCEDURE | VARCHAR2(61) | | Validation procedure |
| STARTUP | VARCHAR2(8) | | Indicates whether the component requires a startup after the upgrade (REQUIRED) or not |
| PARENT_ID | VARCHAR2(30) | | Parent component identifier |
| OTHER_SCHEMAS | VARCHAR2(4000) | | A list of ancillary schema names associated with the component |

**See Also:**

"USER_REGISTRY"

# DBA_REGISTRY_HIERARCHY

DBA_REGISTRY_HIERARCHY displays information about the components loaded into the database, grouped by owner and organized in the component hierarchy.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAMESPACE | VARCHAR2(30) | NOT NULL | Component namespace |
| COMP_ID | VARCHAR2(4000) | | Component identifier |
| VERSION | VARCHAR2(30) | | Component version loaded |
| STATUS | VARCHAR2(11) | | Component status:<br>• INVALID<br>• VALID<br>• LOADING<br>• LOADED<br>• UPGRADING<br>• UPGRADED<br>• DOWNGRADING<br>• DOWNGRADED<br>• REMOVING<br>• REMOVED |
| MODIFIED | VARCHAR2(20) | | Time when the component was last modified |

# DBA_REGISTRY_HISTORY

DBA_REGISTRY_HISTORY provides information about upgrades, downgrades, and critical patch updates that have been performed on the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ACTION_TIME | TIMESTAMP(6) | | The time the upgrade, downgrade, or patch action was completed |
| ACTION | VARCHAR2(30) | | The specific action (for example, UPGRADE or DOWNGRADE) |
| NAMESPACE | VARCHAR2(30) | | The namespace of the components affected (for example, SERVER) |
| VERSION | VARCHAR2(30) | | The version number of the server (for example, 10.2.0.1.0) |
| ID | NUMBER | | Bundle ID |
| COMMENTS | VARCHAR2(255) | | Additional comments about the action taken |

# DBA_REGISTRY_LOG

DBA_REGISTRY_LOG displays operating information about components loaded into the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OPTIME | VARCHAR2(20) | | Operation time |
| NAMESPACE | VARCHAR2(30) | | Component namespace |
| COMP_ID | VARCHAR2(30) | | Component identifier |
| OPERATION | VARCHAR2(11) | | Operation name |
| MESSAGE | VARCHAR2(1000) | | Message |

# DBA_REGISTRY_SQLPATCH

DBA_REGISTRY_SQLPATCH contains information about the SQL patches that have been installed in the database. A SQL patch is a patch that contains SQL scripts which need to be run after OPatch completes. DBA_REGISTRY_SQLPATCH is updated by the datapatch utility. Each row contains information about an installation attempt (apply or roll back) for a given patch.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PATCH_ID | NUMBER | NOT NULL | ID associated with the patch |
| PATCH_UID | NUMBER | NOT NULL | UPI (Universal Patch ID) associated with the patch |
| VERSION | VARCHAR2(20) | NOT NULL | 4 digit version on which the patch was installed (for example, 12.1.0.2) |
| FLAGS | VARCHAR2(10) | | One or more of the following:<br>• U: Patch requires upgrade mode<br>• J: Patch is a JVM patch<br>• F: Patch was installed using -force |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • B: Patch is a bundle patch |
| ACTION | VARCHAR2(15) | NOT NULL | APPLY or ROLLBACK |
| STATUS | VARCHAR2(15) | | Possible values:<br>• SUCCESS: Patch application has completed with no errors<br>• WITH ERRORS: Patch application finished with errors |
| ACTION_TIME | TIMESTAMP(6) | NOT NULL | Timestamp when the install was performed |
| DESCRIPTION | VARCHAR2(100) | | Description of this patch from OPatch metadata |
| BUNDLE_SERIES | VARCHAR2(30) | | If a bundle patch, the series (for example, PSU or DBBP) |
| BUNDLE_ID | NUMBER | | If a bundle patch, ID in the series |
| BUNDLE_DATA | XMLTYPE | | If a bundle patch, the bundledata.xml associated with the patch |
| LOGFILE | VARCHAR2(500) | | Location of the logfile for this apply or rollback attempt |

**See Also:**

- Oracle OPatch User's Guide for Windows and UNIX for more information about OPatch and related patching utilities
- Support note 1585822.1 "Datapatch: Database 12c Post Patch SQL Automation" at My Oracle Support at the following URL for more information about datapatch:

  http://myoraclesupport.com

# DBA_REPAIR_TABLE

DBA_REPAIR_TABLE describes any corruptions found by the DBMS_REPAIR.CHECK_OBJECT procedure. This information is used by the DBMS_REPAIR.FIX_CORRUPT_BLOCKS procedure on execution. To create this view, first run the DBMS_REPAIR.ADMIN_TABLES procedure. To populate the resulting repair table for an object, run the DBMS_REPAIR.CHECK_OBJECT procedure on the object.

**Note:**

The table created by the DBMS_REPAIR.ADMIN_TABLES procedure is called REPAIR TABLE by default. If you specify a different name, this view will have the name you specify, preceded by "DBA_REPAIR_".

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_ID | NUMBER | NOT NULL | Dictionary object number of the object with the corruption |
| TABLESPACE_ID | NUMBER | NOT NULL | Tablespace number of the corrupt object |
| RELATIVE_FILE_ID | NUMBER) | NOT NULL | Relative file number of the corrupt object |
| BLOCK_ID | NUMBER | NOT NULL | Block number of the corruption |
| CORRUPT_TYPE | NUMBER | NOT NULL | Type of corruption encountered |
| SCHEMA_NAME | VARCHAR2(30) | NOT NULL | Schema of the corrupt object |
| OBJECT_NAME | VARCHAR2(30) | NOT NULL | Name of the corrupt object |
| BASEOBJECT_NAME | VARCHAR2(30) | NULL | If the object is an index, the name of its base table |
| PARTITION_NAME | VARCHAR2(30) | NULL | Partition or subpartition name, if applicable |
| CORRUPT_DESCRIPTION | VARCHAR2(2000) | NULL | Description of corruption |
| REPAIR_DESCRIPTION | VARCHAR2(200) | NULL | Description of repair action |
| MARKED_CORRUPT | VARCHAR2(10) | NOT NULL | Whether the block is marked corrupt (TRUE \| FALSE) |
| CHECK_TIMESTAMP | DATE | NOT NULL | Date and time when this row was insert into the repair table |
| FIX_TIMESTAMP | DATE | NULL | Date and time when the block was modified by the FIX_CORRUPT_BLOCKS procedure, if applicable |
| REFORMAT_TIMESTAMP | DATE | NULL | Reserved for future use |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_REPAIR package

# DBA_REPL_DBNAME_MAPPING

DBA_REPL_DBNAME_MAPPING provides details about the database name mapping in replication. Its columns are the same as those in ALL_REPL_DBNAME_MAPPING.

**See Also:**

"ALL_REPL_DBNAME_MAPPING"

# DBA_RESOURCE_INCARNATIONS

DBA_RESOURCE_INCARNATIONS lists all resource incarnations that are running or eligible for HA status notification.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RESOURCE_TYPE | VARCHAR2(30) | NOT NULL | Type of resource |
| RESOURCE_NAME | VARCHAR2(256) | | Name of resource |
| DB_UNIQUE_NAME | VARCHAR2(30) | NOT NULL | Database unique name |
| DB_DOMAIN | VARCHAR2(128) | NOT NULL | Database domain |
| INSTANCE_NAME | VARCHAR2(30) | NOT NULL | Name of instance at which resource is located |
| HOST_NAME | VARCHAR2(512) | | Name of host at which resource is located |
| STARTUP_TIME | TIMESTAMP(9) WITH TIME ZONE | | Resource startup date and time |

# DBA_RESUMABLE

DBA_RESUMABLE displays all resumable statements executed in the system.

### Related View

USER_RESUMABLE displays the resumable statements executed by the current user. This view does not display the USER_ID column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_ID | NUMBER | | User ID Number of the Resumable Statement Owner |
| SESSION_ID | NUMBER | | Session Identifier of the Resumable Statement |
| INSTANCE_ID | NUMBER | | Instance Number of the Resumable Statement |
| COORD_INSTANCE_ID | NUMBER | | Instance Number on which the Parallel Coordinator is Running |
| COORD_SESSION_ID | NUMBER | | Session Identifier of the Parallel Coordinator |
| STATUS | VARCHAR2(9) | | Status of the resumable statement:<br>• RUNNING<br>• SUSPENDED<br>• TIMEOUT<br>• ERROR<br>• ABORTED |
| TIMEOUT | NUMBER | | Timeout of the resumable statement |
| START_TIME | VARCHAR2(20) | | Start time of the resumable statement |
| SUSPEND_TIME | VARCHAR2(20) | | Last time the resumable statement was suspended (initialized to NULL) |
| RESUME_TIME | VARCHAR2(20) | | Last time the suspended resumable statement was resumed (initialized to NULL) |
| NAME | VARCHAR2(4000) | | Name given in the resumable clause of the resumable statement |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SQL_TEXT | VARCHAR2(1000) | | Resumable statement, selected from the V$SQL view |
| ERROR_NUMBER | NUMBER | | Error code of the last correctable error. When STATUS is set to RUNNING, its value will be 0. |
| ERROR_PARAMETER1 | VARCHAR2(80) | | First parameter for the error message (NULL if no error) |
| ERROR_PARAMETER2 | VARCHAR2(80) | | Second parameter for the error message (NULL if no error) |
| ERROR_PARAMETER3 | VARCHAR2(80) | | Third parameter for the error message (NULL if no error) |
| ERROR_PARAMETER4 | VARCHAR2(80) | | Forth parameter for the error message (NULL if no error) |
| ERROR_PARAMETER5 | VARCHAR2(80) | | Fifth parameter for the error message (NULL if no error) |
| ERROR_MSG | VARCHAR2(4000) | | Error message corresponding to ERROR_NUMBER. It will be NULL when ERROR_NUMBER is 0. |

**See Also:**

"USER_RESUMABLE"

# DBA_REWRITE_EQUIVALENCES

DBA_REWRITE_EQUIVALENCES describes all rewrite equivalences in the database. Its columns are the same as those in ALL_REWRITE_EQUIVALENCES.

**See Also:**

"ALL_REWRITE_EQUIVALENCES"

# DBA_RGROUP

DBA_RGROUP displays all refresh groups.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REFGROUP | NUMBER | | Internal identifier of the refresh group |
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object in the refresh group |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object in the refresh group |
| IMPLICIT_DESTROY | VARCHAR2(1) | | Indicates whether the refresh group is destroyed when its last item is removed (Y) or not (N) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PUSH_DEFERRED_RPC | VARCHAR2(1) | | Indicates whether changes are pushed from the snapshot to the master before refresh (Y) or not (N) |
| REFRESH_AFTER_ERRORS | VARCHAR2(1) | | Indicates whether to proceed with refresh despite errors when pushing deferred RPCs (Y) or not (N) |
| ROLLBACK_SEG | VARCHAR2(128) | | Name of the rollback segment to use while refreshing |
| JOB | NUMBER | NOT NULL | Identifier of the job used to refresh the group automatically |
| PURGE_OPTION | NUMBER(38) | | Method for purging the transaction queue after each push. 1 indicates quick purge option; 2 indicates precise purge option |
| PARALLELISM | NUMBER(38) | | Level of parallelism for transaction propagation |
| HEAP_SIZE | NUMBER(38) | | Size of the heap |

# DBA_ROLE_PRIVS

DBA_ROLE_PRIVS describes the roles granted to all users and roles in the database.

### Related View

USER_ROLE_PRIVS describes the roles granted to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Name of the user or role receiving the grant |
| GRANTED_ROLE | VARCHAR2(128) | | Granted role name |
| ADMIN_OPTION | VARCHAR2(3) | | Indicates whether the grant was with the ADMIN OPTION (YES) or not (NO) |
| DELEGATE_OPTION[1] | VARCHAR2(3) | | Indicates whether the grant was with the DELEGATE OPTION (YES) or not (NO) |
| DEFAULT_ROLE | VARCHAR2(3) | | Indicates whether the role is designated as a DEFAULT ROLE for the user (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the role was granted commonly (CONTAINER=ALL was used)<br>• NO if the role was granted locally (CONTAINER=ALL was not used) |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"USER_ROLE_PRIVS"

# DBA_ROLES

DBA_ROLES describes all roles in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROLE | VARCHAR2(128) | NOT NULL | Name of the role |
| PASSWORD_REQUIRED | VARCHAR2(8) | | This column is deprecated in favor of the AUTHENTICATION_TYPE column |
| AUTHENTICATION_TYPE | VARCHAR2(11) | | Indicates the authentication mechanism for the role:<br>• NONE - CREATE ROLE *role1*;<br>• EXTERNAL - CREATE ROLE *role2* IDENTIFIED EXTERNALLY;<br>• GLOBAL - CREATE ROLE *role3* IDENTIFIED GLOBALLY;<br>• APPLICATION - CREATE ROLE *role4* IDENTIFIED USING *schema.package*;<br>• PASSWORD - CREATE ROLE *role5* IDENTIFIED BY *role5*; |
| COMMON | VARCHAR2(3) | | Indicates whether a given role is common. Possible values:<br>• YES if the role is common<br>• NO if the role is local (not common) |
| ORACLE_MAINTAINED | VARCHAR2(1) | | Denotes whether the role was created, and is maintained, by Oracle-supplied scripts (such as catalog.sql or catproc.sql). A role for which this column has the value Y must not be changed in any way except by running an Oracle-supplied script. |

# DBA_ROLLBACK_SEGS

DBA_ROLLBACK_SEGS describes rollback segments.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SEGMENT_NAME | VARCHAR2(30) | NOT NULL | Name of the rollback segment |
| OWNER | VARCHAR2(6) | | Owner of the rollback segment:<br>• PUBLIC<br>• SYS |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace containing the rollback segment |
| SEGMENT_ID | NUMBER | NOT NULL | ID number of the rollback segment |
| FILE_ID | NUMBER | NOT NULL | File identifier number of the file containing the segment head |
| BLOCK_ID | NUMBER | NOT NULL | ID number of the block containing the segment header |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INITIAL_EXTENT | NUMBER | | Initial extent size in bytes |
| NEXT_EXTENT | NUMBER | | Secondary extent size in bytes |
| MIN_EXTENTS | NUMBER | NOT NULL | Minimum number of extents |
| MAX_EXTENTS | NUMBER | NOT NULL | Maximum number of extent |
| PCT_INCREASE | NUMBER | | Percent increase for extent size |
| STATUS | VARCHAR2(16) | | Rollback segment status:<br>• OFFLINE<br>• ONLINE<br>• NEEDS RECOVERY<br>• PARTLY AVAILABLE<br>• UNDEFINED |
| INSTANCE_NUM | VARCHAR2(40) | | Rollback segment owning Oracle Real Application Clusters instance number |
| RELATIVE_FNO | NUMBER | NOT NULL | Relative file number of the segment header |

# DBA_ROLLING_DATABASES

DBA_ROLLING_DATABASES lists all the databases eligible for configuration with rolling operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RDBID | NUMBER | | Rolling operation database identifier |
| DBID | NUMBER | | Oracle database identifier |
| DBUN | VARCHAR2(128) | | Database unique name |
| ROLE | VARCHAR2(8) | | Database role |
| OPEN_MODE | VARCHAR2(15) | | Open mode information |
| PARTICIPANT | VARCHAR2(3) | | Indicates whether the database is participating in the rolling operation (YES) or not (NO) |
| VERSION | VARCHAR2(128) | | RDBMS version number |
| ENGINE_STATUS | VARCHAR2(14) | | Running status of the MRP-recovery or LSP-apply process |
| RAC | VARCHAR2(3) | | Indicates whether the database is an Oracle Real Application Clusters (Oracle RAC) database |
| UPDATE_PROGRESS | VARCHAR2(11) | | Upgrade status of the system catalog |
| PROD_RSCN | VARCHAR(40) | | Resetlogs SCN at which redo is currently being produced |
| PROD_RID | VARCHAR(40) | | Resetlogs ID at which redo is currently being produced |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROD_SCN | VARCHAR(40) | | Last SCN at which redo was produced |
| REDO_SOURCE | VARCHAR2(128) | | Database unique name of the producer of redo being consumed |
| CONS_RSCN | VARCHAR(40) | | Resetlogs SCN at which redo is currently being consumed |
| CONS_RID | VARCHAR(40) | | Resetlogs ID at which redo is currently being consumed |
| CONS_SCN | VARCHAR(40) | | Last SCN at which redo was consumed |
| UPDATE_TIME | TIMESTAMP(6) | | Time of the last record update |

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about rolling operations.

# DBA_ROLLING_EVENTS

DBA_ROLLING_EVENTS lists all the events reported from the DBMS_ROLLING PL/SQL package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVENTID | NUMBER | | Event identifier which identifies event order |
| EVENT_TIME | TIMESTAMP(6) | | Time associated with the event |
| TYPE | VARCHAR2(7) | | Type of event: INFO, NOTICE, WARNING, or ERROR |
| MESSAGE | VARCHAR2(256) | | Text describing the event details |
| STATUS | NUMBER | | Status code associated with an event |
| INSTID | NUMBER | | Instruction ID associated with an event |
| REVISION | NUMBER | | Plan revision number associated with an event |

**See Also:**

- *Oracle Data Guard Concepts and Administration* for more information about rolling operations.
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ROLLING package

# DBA_ROLLING_PARAMETERS

DBA_ROLLING_PARAMETERS lists the available parameters of the DBMS_ROLLING PL/SQL package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCOPE | VARCHAR2(128) | | Database unique name associated with a parameter |
| TYPE | VARCHAR2(7) | | Type of parameter |
| NAME | VARCHAR2(32) | | Name of the parameter |
| DESCRIPTION | VARCHAR2(256) | | Description of the parameter |
| CURVAL | VARCHAR2(256) | | Current value of the parameter |
| LSTVAL | VARCHAR2(256) | | Prior value of the parameter |
| DEFVAL | VARCHAR2(256) | | Default value of the parameter |
| MINVAL | NUMBER | | Minimum value of the parameter |
| MAXVAL | NUMBER | | Maximum value of the parameter |

**See Also:**

- *Oracle Data Guard Concepts and Administration* for more information about rolling operations.
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ROLLING package

# DBA_ROLLING_PLAN

DBA_ROLLING_PLAN displays the instructions which constitute the active upgrade plan. Each row in DBA_ROLLING_PLAN identifies a specific instruction scheduled to execute at a specific database. Instructions are created as a result of successful calls to the DBMS_ROLLING.BUILD_PLAN procedure.

During execution, groups of instructions are scheduled in batches to execute at remote databases. Groups of instructions are guaranteed to complete in BATCHID order.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REVISION | NUMBER | | Plan revision number associated with an instruction |
| BATCHID | NUMBER | | Identifier for a batch of instructions which are requested together |
| INSTID | NUMBER | | Identifier for a single instruction |
| SOURCE | VARCHAR2(128) | | Database unique name where an instruction executes |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TARGET | VARCHAR2(128) | | The site where a given instruction will execute |
| PHASE | VARCHAR2(14) | | rolling operation phase in which an instruction executes |
| STATUS | VARCHAR2(7) | | Scheduling status of the instruction |
| PROGRESS | VARCHAR2(10) | | Execution progress of the instruction |
| DESCRIPTION | VARCHAR2(256) | | Description of the instruction |
| EXEC_STATUS | NUMBER | | Status code returned from instruction execution |
| EXEC_INFO | VARCHAR2(256) | | Supplemental information obtained during instruction execution |
| EXEC_TIME | TIMESTAMP(6) | | Time of instruction execution |
| FINISH_TIME | TIMESTAMP(6) | | Time of instruction completion |

**See Also:**

- *Oracle Data Guard Concepts and Administration* for more information about rolling operations.
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ROLLING package

## DBA_ROLLING_STATISTICS

DBA_ROLLING_STATISTICS provides a list of rolling operation statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(256) | | Name of the statistic |
| VALUE | VARCHAR2(256) | | Value of the statistic |
| UPDATE_TIME | TIMESTAMP(6) | | Time of last update |

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about rolling operations.

## DBA_ROLLING_STATUS

DBA_ROLLING_STATUS displays the overall status of the rolling operation.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REVISION | NUMBER | | Revision number of the current upgrade plan |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATUS | VARCHAR2(12) | | Readiness of the facility to begin or resume the rolling operation |
| PHASE | VARCHAR2(14) | | Current phase of the plan |
| NEXT_INSTRUCTION | NUMBER | | Instruction ID of the next pending instruction |
| REMAINING_INSTRUCTIONS | NUMBER | | Number of remaining instructions to execute in the plan |
| COORDINATOR_INSTANCE | NUMBER | | Instance number from which the rolling operation is being coordinated |
| COORDINATOR_PID | NUMBER | | Process PID in which the rolling operation is being coordinated |
| ORIGINAL_PRIMARY | VARCHAR2(128) | | Database unique name of the original primary |
| FUTURE_PRIMARY | VARCHAR2(128) | | Database unique name of the future primary |
| TOTAL_DATABASES | NUMBER | | Number of total databases eligible to participate in the rolling operation |
| PARTICIPATING_DATABASES | NUMBER | | Number of databases configured to participate in the rolling operation |
| INIT_TIME | TIMESTAMP(6) | | Time of the last call to DBMS_ROLLING.INIT_PLAN |
| BUILD_TIME | TIMESTAMP(6) | | Time of the last call to DBMS_ROLLING.BUILD |
| START_TIME | TIMESTAMP(6) | | Time of the last call to DBMS_ROLLING.START_UPGRADE |
| SWITCH_TIME | TIMESTAMP(6) | | Time of the last call to DBMS_ROLLING.SWITCHOVER |
| FINISH_TIME | TIMESTAMP(6) | | Time of the last call to DBMS_ROLLING.FINISH |

**See Also:**

- *Oracle Data Guard Concepts and Administration* for more information about rolling operations.
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ROLLING package

# DBA_ROLLING_UNSUPPORTED

DBA_ROLLING_UNSUPPORTED displays the schemas, tables, and columns in those tables that contain unsupported data types for a rolling upgrade operation for a logical standby database using the DBMS_ROLLING PL/SQL package. Use this view before you perform a rolling upgrade using DBMS_ROLLING to determine what is unsupported.

The data pertains to the container in which the view is queried.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(30) | | Schema name of the unsupported column |
| TABLE_NAME | VARCHAR2(30) | | Name of the table that the unsupported column belongs to |
| COLUMN_NAME | VARCHAR2(30) | | Name of the unsupported column |
| ATTRIBUTES | VARCHAR2(39) | | If not a data type issue, displays the reason why the table is unsupported |
| DATA_TYPE | VARCHAR2(32) | | Data type of the unsupported column |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

A rolling upgrade using `DBMS_ROLLING` supports more object types than a manual rolling upgrade using transient logical standby databases

**See Also:**

- "DBA_LOGSTDBY_UNSUPPORTED" for more information about determining unsupported data types for a manual rolling upgrade operation using transient logical standby databases
- *Oracle Data Guard Concepts and Administration* for more information about rolling operations
- *Oracle Data Guard Concepts and Administration* for more information about unsupported tables for rolling upgrade operations
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_ROLLING` package

# DBA_RSRC_CATEGORIES

`DBA_RSRC_CATEGORIES` displays all resource consumer group categories.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | | Name of the consumer group category |
| COMMENTS | VARCHAR2(2000) | | Text comment on the consumer group category |
| STATUS | VARCHAR2(128) | | Indicates whether the consumer group category is part of the pending area (`PENDING`) or not (NULL) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MANDATORY | VARCHAR2(3) | | Indicates whether the consumer group category is mandatory (YES) or not (NO) |

# DBA_RSRC_CONSUMER_GROUP_PRIVS

DBA_RSRC_CONSUMER_GROUP_PRIVS displays information about all resource consumer groups and the users and roles assigned to them. The grant referred to in this view and the related view is the grant of the SWITCH_CONSUMER_GROUP object privilege, which is granted using the DBMS_RESOURCE_MANAGER_PRIVS package. This privilege is not granted through the GRANT SQL statement.

### Related View

USER_RSRC_CONSUMER_GROUP_PRIVS displays information about the resource consumer groups to which the current user is assigned. This view does not display the GRANTEE column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | NOT NULL | User or role receiving the grant |
| GRANTED_GROUP | VARCHAR2(128) | | Granted consumer group name |
| GRANT_OPTION | VARCHAR2(3) | | Indicates whether the grant was with the GRANT option (YES) or not (NO) |
| INITIAL_GROUP | VARCHAR2(3) | | Indicates whether the consumer group is designated as the default for this user or role (YES) or not (NO) |

**See Also:**

- "USER_RSRC_CONSUMER_GROUP_PRIVS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_RESOURCE_MANAGER_PRIVS package

# DBA_RSRC_CONSUMER_GROUPS

DBA_RSRC_CONSUMER_GROUPS displays information about all resource consumer groups in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CONSUMER_GROUP_ID | NUMBER | NOT NULL | ID of the consumer group |
| CONSUMER_GROUP | VARCHAR2(128) | | Name of the consumer group |
| CPU_METHOD | VARCHAR2(128) | | CPU resource allocation method for the consumer group |
| MGMT_METHOD | VARCHAR2(128) | | Resource allocation method for the consumer group |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INTERNAL_USE | VARCHAR2(3) | | Indicates whether the consumer group is for internal use only (YES) or not (NO) |
| COMMENTS | VARCHAR2(2000) | | Text comment on the consumer group |
| CATEGORY | VARCHAR2(128) | | Category of the consumer group |
| STATUS | VARCHAR2(128) | | Indicates whether the consumer group is part of the pending area (PENDING) or not (NULL) |
| MANDATORY | VARCHAR2(3) | | Indicates whether the consumer group is mandatory (YES) or not (NO) |

# DBA_RSRC_GROUP_MAPPINGS

DBA_RSRC_GROUP_MAPPINGS displays the mapping between session attributes and consumer groups in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ATTRIBUTE | VARCHAR2(128) | | Session attribute to match |
| VALUE | VARCHAR2(128) | | Attribute value |
| CONSUMER_GROUP | VARCHAR2(128) | | Target consumer group name |
| STATUS | VARCHAR2(128) | | Indicates whether the consumer group is part of the pending area (PENDING) or not (NULL) |

# DBA_RSRC_IO_CALIBRATE

DBA_RSRC_IO_CALIBRATE displays I/O calibration results for the latest calibration run.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| START_TIME | TIMESTAMP(6) | | Start time of the most recent I/O calibration |
| END_TIME | TIMESTAMP(6) | | End time of the most recent I/O calibration |
| MAX_IOPS | NUMBER | | Maximum number of data block read requests that can be sustained per second |
| MAX_MBPS | NUMBER | | Maximum megabytes per second of maximum-sized read requests that can be sustained |
| MAX_PMBPS | NUMBER | | Maximum megabytes per second of large I/O requests that can be sustained by a single process |
| LATENCY | NUMBER | | Latency for data block read requests |
| NUM_PHYSICAL_DISKS | NUMBER | | Number of physical disks in the storage subsystem (as specified by the user) |

# DBA_RSRC_MANAGER_SYSTEM_PRIVS

DBA_RSRC_MANAGER_SYSTEM_PRIVS displays information about all the users and roles that have been granted the ADMINISTER_RESOURCE_MANAGER system privilege, which is granted using the DBMS_RESOURCE_MANAGER_PRIVS package. This privilege is not granted through the GRANT SQL statement.

### Related View

USER_RSRC_MANAGER_SYSTEM_PRIVS displays information about the users who are granted system privileges for the DBMS_RESOURCE_MANAGER package. This view does not display the GRANTEE column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | NOT NULL | User or role receiving the grant |
| PRIVILEGE | VARCHAR2(40) | NOT NULL | Name of the system privilege |
| ADMIN_OPTION | VARCHAR2(3) | | Indicates whether the grant was with the ADMIN option (YES) or not (NO) |

**See Also:**

- "USER_RSRC_MANAGER_SYSTEM_PRIVS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_RESOURCE_MANAGER package

# DBA_RSRC_MAPPING_PRIORITY

DBA_RSRC_MAPPING_PRIORITY displays information about all consumer group mapping attribute priorities.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ATTRIBUTE | VARCHAR2(128) | | Session attribute |
| PRIORITY | NUMBER | | Priority (1 is the highest) |
| STATUS | VARCHAR2(128) | | Indicates whether the consumer group is part of the pending area (PENDING) or not (NULL) |

# DBA_RSRC_PLAN_DIRECTIVES

DBA_RSRC_PLAN_DIRECTIVES displays information about all resource plan directives in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PLAN | VARCHAR2(128) | | Name of the plan to which the directive belongs |
| GROUP_OR_SUBPLAN | VARCHAR2(128) | | Name of the consumer group or subplan referred to |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE | VARCHAR2(14) | | Indicates whether GROUP_OR_SUBPLAN refers to a consumer group (CONSUMER_GROUP) or a plan (PLAN) |
| CPU_P1 | NUMBER | | This column is deprecated. Use the MGMT_P1 column instead. |
| CPU_P2 | NUMBER | | This column is deprecated. Use the MGMT_P2 column instead. |
| CPU_P3 | NUMBER | | This column is deprecated. Use the MGMT_P3 column instead. |
| CPU_P4 | NUMBER | | This column is deprecated. Use the MGMT_P4 column instead. |
| CPU_P5 | NUMBER | | This column is deprecated. Use the MGMT_P5 column instead.. |
| CPU_P6 | NUMBER | | This column is deprecated. Use the MGMT_P6 column instead. |
| CPU_P7 | NUMBER | | This column is deprecated. Use the MGMT_P7 column instead. |
| CPU_P8 | NUMBER | | This column is deprecated. Use the MGMT_P8 column instead.. |
| MGMT_P1 | NUMBER | | Resource allocation at level 1. |
| MGMT_P2 | NUMBER | | Resource allocation at level 2. |
| MGMT_P3 | NUMBER | | Resource allocation at level 3. |
| MGMT_P4 | NUMBER | | Resource allocation at level 4. |
| MGMT_P5 | NUMBER | | Resource allocation at level 5. |
| MGMT_P6 | NUMBER | | Resource allocation at level 6. |
| MGMT_P7 | NUMBER | | Resource allocation at level 7. |
| MGMT_P8 | NUMBER | | Resource allocation at level 8. |
| ACTIVE_SESS_POOL_P1 | NUMBER | | Maximum number of calls this consumer group can run concurrently |
| QUEUEING_P1 | NUMBER | | Timeout in seconds for waits in the Active Session Limit queue |
| PARALLEL_TARGET_PERCENTAGE | NUMBER | | This column is deprecated. Use the PARALLEL_SERVER_LIMIT column instead. |
| PARALLEL_DEGREE_LIMIT_P1 | NUMBER | | Sessions in this consumer group are limited to this maximum degree of parallelism for all parallel operations |
| SWITCH_GROUP | VARCHAR2(128) | | Group to switch to once the switch time is reached |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SWITCH_FOR_CALL | VARCHAR2(5) | | Indicates whether to switch back to the initial consumer group once the top call has completed (TRUE) or not (FALSE) |
| SWITCH_TIME | NUMBER | | Amount of run time (in seconds) within a group before the session is automatically switched. As with other switch directives, if SWITCH_FOR_CALL is TRUE, the run time is accumulated from the start of a call. Otherwise, the run time is accumulated for the length of the session. |
| SWITCH_IO_MEGABYTES | NUMBER | | The maximum megabytes of I/O within a group that will trigger the action specified by SWITCH_GROUP. As with other switch directives, if SWITCH_FOR_CALL is TRUE, the maximum megabytes of I/O is accumulated from the start of a call. Otherwise, the maximum megabytes of I/O is accumulated for the length of the session. |
| SWITCH_IO_REQS | NUMBER | | The maximum I/O requests within a group that will trigger the action specified by SWITCH_GROUP. As with other switch directives, if SWITCH_FOR_CALL is TRUE, the maximum I/O requests is accumulated from the start of a call. Otherwise, the maximum I/O requests is accumulated for the length of the session. |
| SWITCH_ESTIMATE | VARCHAR2(5) | | Indicates whether estimated execution time should be used for switch criteria (TRUE) or not (FALSE) |
| MAX_EST_EXEC_TIME | NUMBER | | Maximum estimated execution time |
| UNDO_POOL | NUMBER | | Undo pool size for the consumer group |
| MAX_IDLE_TIME | NUMBER | | Maximum idle time for the session |
| MAX_IDLE_BLOCKER_TIME | NUMBER | | Maximum idle time for the session when blocking other sessions |
| MAX_UTILIZATION_LIMIT | NUMBER | | This column is deprecated. Use UTILIZATION_LIMIT instead. |
| PARALLEL_QUEUE_TIMEOUT | NUMBER | | Time (in seconds) that a query may remain in the parallel statement queue for the consumer group before it is removed and terminated with error ORA-07454 |
| SWITCH_TIME_IN_CALL | NUMBER | | This column is deprecated. Specify the time in the SWITCH_TIME directive and set SWITCH_FOR_CALL to TRUE. |
| SWITCH_IO_LOGICAL | NUMBER | | The number of logical I/Os that will trigger the action specified by SWITCH_GROUP. As with other switch directives, if SWITCH_FOR_CALL is TRUE, the number of logical I/Os is accumulated from the start of a call. Otherwise, the number of |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | logical IOs is accumulated for the length of the session. |
| SWITCH_ELAPSED_TIME | NUMBER | | The elapsed time that will trigger the action specified by SWITCH_GROUP. As with other switch directives, if SWITCH_FOR_CALL is TRUE, the elapsed time is accumulated from the start of a call. Otherwise, the elapsed time is accumulated for the length of the session. |
| PARALLEL_SERVER_LIMIT | NUMBER | | Maximum percentage of the parallel target used before queuing subsequent parallel queries |
| UTILIZATION_LIMIT | NUMBER | | Maximum resource utilization allowed, expressed in percentage |
| PARALLEL_STMT_CRITICAL | VARCHAR2(12) | | Indicates whether parallel statements from this consumer group are critical and should not be queued:<br>• FALSE - The consumer group is non-critical<br>• BYPASS QUEUE - The consumer group is marked critical, and parallel statements from this conumser group will bypass the parallel statement queue |
| COMMENTS | VARCHAR2(2000) | | Text comment on the plan directive |
| STATUS | VARCHAR2(128) | | Indicates whether the plan directive is part of the pending area (PENDING) or not (NULL).<br>**Note:** PDB resource plans must be single-level, they cannot contain subplans, and they must have 8 or fewer consumer groups. If a resource plan is imported into a PDB and it violates any of these PDB requirements, then the import will automatically convert the resource plan to a compliant version. The original, unmodified resource plan will be stored with a STATUS of LEGACY. |
| MANDATORY | VARCHAR2(3) | | Indicates whether the plan directive is mandatory (YES) or not (NO). Mandatory plans cannot be deleted. |

**See Also:**

- *Oracle Database Administrator's Guide* for information on resource plans in general
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_RESOURCE_MANAGER_PRIVS package

# DBA_RSRC_PLANS

DBA_RSRC_PLANS displays information about all resource plans in the database. For a list of currently active plans, see "V$RSRC_PLAN".

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PLAN_ID | NUMBER | NOT NULL | ID number of the resource plan |
| PLAN | VARCHAR2(128) | | Name of the resource plan |
| NUM_PLAN_DIRECTIVES | NUMBER | | Number of plan directives for the plan |
| CPU_METHOD | VARCHAR2(128) | | CPU resource allocation method for the plan |
| MGMT_METHOD | VARCHAR2(128) | | Resource allocation method for the plan |
| ACTIVE_SESS_POOL_MTH | VARCHAR2(128) | | Active session pool resource allocation method for the plan |
| PARALLEL_DEGREE_LIMIT_MTH | VARCHAR2(128) | | Parallel degree limit resource allocation method for the plan |
| QUEUING_MTH | VARCHAR2(128) | | Queuing resource allocation method for the plan |
| SUB_PLAN | VARCHAR2(3) | | Indicates whether the plan is a subplan (YES) or not (NO) |
| COMMENTS | VARCHAR2(2000) | | Text comment on the plan |
| STATUS | VARCHAR2(128) | | Indicates whether the plan is part of the pending area (PENDING) or not (NULL) |
| MANDATORY | VARCHAR2(3) | | Indicates whether the plan is mandatory (YES) or not (NO). Mandatory plans cannot be deleted. |

**See Also:**

- *Oracle Database Administrator's Guide* for information on resource plans in general
- *Oracle Database PL/SQL Packages and Types Reference* for more information on creating resource plans with the DBMS_RESOURCE_MANAGER package

# DBA_RULE_SET_RULES

DBA_RULE_SET_RULES describes the rules in all rule sets in the database. Its columns are the same as those in ALL_RULE_SET_RULES.

**See Also:**

"ALL_RULE_SET_RULES"

# DBA_RULE_SETS

DBA_RULE_SETS describes all rule sets in the database. Its columns are the same as those in ALL_RULE_SETS.

**See Also:**

"ALL_RULE_SETS"

## DBA_RULES

DBA_RULES describes all rules in the database. Its columns are the same as those in ALL_RULES.

**See Also:**

"ALL_RULES"

## DBA_SCHEDULER_CHAIN_RULES

DBA_SCHEDULER_CHAIN_RULES displays information about the rules for all chains in the database. Its columns are the same as those in ALL_SCHEDULER_CHAIN_RULES.

**See Also:**

"ALL_SCHEDULER_CHAIN_RULES"

## DBA_SCHEDULER_CHAIN_STEPS

DBA_SCHEDULER_CHAIN_STEPS displays information about the defined steps of all chains in the database. Its columns are the same as those in ALL_SCHEDULER_CHAIN_STEPS.

**See Also:**

"ALL_SCHEDULER_CHAIN_STEPS"

## DBA_SCHEDULER_CHAINS

DBA_SCHEDULER_CHAINS displays information about all chains in the database. Its columns are the same as those in ALL_SCHEDULER_CHAINS.

**See Also:**

"ALL_SCHEDULER_CHAINS"

## DBA_SCHEDULER_CREDENTIALS

DBA_SCHEDULER_CREDENTIALS displays information about all credentials in the database. Its columns are the same as those in ALL_SCHEDULER_CREDENTIALS.

**Note:**

This view is deprecated in favor of the DBA_CREDENTIALS view. Oracle recommends that you use DBA_CREDENTIALS instead. DBA_SCHEDULER_CREDENTIALS is retained for backward compatibility only.

**See Also:**

- "DBA_CREDENTIALS"
- "ALL_SCHEDULER_CREDENTIALS"

## DBA_SCHEDULER_DB_DESTS

DBA_SCHEDULER_DB_DESTS displays information about all destination objects in the database pointing to remote databases. Its columns are the same as those in ALL_SCHEDULER_DB_DESTS.

**See Also:**

"ALL_SCHEDULER_DB_DESTS"

## DBA_SCHEDULER_DESTS

DBA_SCHEDULER_DESTS displays information about all destination objects for jobs in the database. Its columns are the same as those in ALL_SCHEDULER_DESTS.

**See Also:**

"ALL_SCHEDULER_DESTS"

## DBA_SCHEDULER_EXTERNAL_DESTS

DBA_SCHEDULER_EXTERNAL_DESTS displays information about all destination objects in the database pointing to remote agents. Its columns are the same as those in ALL_SCHEDULER_EXTERNAL_DESTS.

**See Also:**

"ALL_SCHEDULER_EXTERNAL_DESTS"

## DBA_SCHEDULER_FILE_WATCHERS

DBA_SCHEDULER_FILE_WATCHERS displays information about all scheduler file watch requests in the database. Its columns are the same as those in ALL_SCHEDULER_FILE_WATCHERS.

**See Also:**

"ALL_SCHEDULER_FILE_WATCHERS"

## DBA_SCHEDULER_GLOBAL_ATTRIBUTE

DBA_SCHEDULER_GLOBAL_ATTRIBUTE displays the values of all scheduler attributes in the database (for example, DEFAULT_TIMEZONE and CURRENT_OPEN_WINDOW). Its columns are the same as those in ALL_SCHEDULER_GLOBAL_ATTRIBUTE.

**See Also:**

"ALL_SCHEDULER_GLOBAL_ATTRIBUTE"

## DBA_SCHEDULER_GROUP_MEMBERS

DBA_SCHEDULER_GROUP_MEMBERS displays information about the members of all Scheduler object groups in the database. Its columns are the same as those in ALL_SCHEDULER_GROUP_MEMBERS.

**See Also:**

"ALL_SCHEDULER_GROUP_MEMBERS"

## DBA_SCHEDULER_GROUPS

DBA_SCHEDULER_GROUPS displays information about all Scheduler object groups in the database. Its columns are the same as those in ALL_SCHEDULER_GROUPS.

**See Also:**

"ALL_SCHEDULER_GROUPS"

## DBA_SCHEDULER_JOB_ARGS

DBA_SCHEDULER_JOB_ARGS displays information about the arguments of all Scheduler jobs in the database. Its columns are the same as those in ALL_SCHEDULER_JOB_ARGS.

**See Also:**

"ALL_SCHEDULER_JOB_ARGS"

## DBA_SCHEDULER_JOB_CLASSES

DBA_SCHEDULER_JOB_CLASSES displays information about all Scheduler job classes in the database. Its columns are the same as those in ALL_SCHEDULER_JOB_CLASSES.

**See Also:**

"ALL_SCHEDULER_JOB_CLASSES"

## DBA_SCHEDULER_JOB_DESTS

DBA_SCHEDULER_JOB_DESTS displays information about the state of all jobs in the database at each of their destinations. Its columns are the same as those in ALL_SCHEDULER_JOB_DESTS.

**See Also:**

"ALL_SCHEDULER_JOB_DESTS"

## DBA_SCHEDULER_JOB_LOG

DBA_SCHEDULER_JOB_LOG displays log information for all Scheduler jobs in the database. Its columns are the same as those in ALL_SCHEDULER_JOB_LOG.

**See Also:**

"ALL_SCHEDULER_JOB_LOG"

## DBA_SCHEDULER_JOB_ROLES

DBA_SCHEDULER_JOB_ROLES displays information about all Scheduler jobs in the database by database role.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the Scheduler job |
| JOB_NAME | VARCHAR2(128) | NOT NULL | Name of the Scheduler job |
| JOB_SUBNAME | VARCHAR2(128) | | Subname of the Scheduler job (for a job running a chain step) |
| JOB_CREATOR | VARCHAR2(128) | | Creator of the Scheduler job |
| DATABASE_ROLE | VARCHAR2(16) | | Name of the database role |
| PROGRAM_OWNER | VARCHAR2(4000) | | Owner of the program associated with the job |
| PROGRAM_NAME | VARCHAR2(4000) | | Name of the program associated with the job |
| JOB_TYPE | VARCHAR2(16) | | Inline job action type:<br>• PLSQL_BLOCK<br>• STORED_PROCEDURE<br>• EXECUTABLE<br>• CHAIN |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ACTION | VARCHAR2(4000) | | Inline job action |
| JOB_CLASS | VARCHAR2(128) | | Name of the job class associated with the job |
| SCHEDULE_OWNER | VARCHAR2(4000) | | Owner of the schedule that the job uses (can be a window or a window group) |
| SCHEDULE_NAME | VARCHAR2(4000) | | Name of the schedule that the job uses (can be a window or a window group) |
| SCHEDULE_TYPE | VARCHAR2(12) | | Type of the schedule that the job uses:<br>• IMMEDIATE - Start date and repeat interval are NULL<br>• ONCE - Repeat interval is NULL<br>• PLSQL - PL/SQL expression used as schedule<br>• CALENDAR - Oracle calendaring expression used as schedule<br>• EVENT - Event schedule<br>• NAMED - Named schedule<br>• WINDOW - Window used as schedule<br>• WINDOW_GROUP - Window group used as schedule |
| START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Original scheduled start date of the job (for an inline schedule) |
| REPEAT_INTERVAL | VARCHAR2(4000) | | Inline schedule PL/SQL expression or calendar string |
| END_DATE | TIMESTAMP(6) WITH TIME ZONE | | Date after which the job will no longer run (for an inline schedule) |
| LAST_START_DATE | TIMESTAMP(6) WITH TIME ZONE | | Last date on which the job ran |
| ENABLED | VARCHAR2(5) | | Indicates whether the job is enabled (TRUE) or disabled (FALSE) |
| STATE | VARCHAR2(15) | | Current state of the job:<br>• DISABLED<br>• RETRY SCHEDULED<br>• SCHEDULED<br>• RUNNING<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• REMOTE<br>• SUCCEEDED<br>• CHAIN_STALLED |
| COMMENTS | VARCHAR2(240) | | Comments on the job |

## DBA_SCHEDULER_JOB_RUN_DETAILS

DBA_SCHEDULER_JOB_RUN_DETAILS displays log run details for all Scheduler jobs in the database. Its columns are the same as those in ALL_SCHEDULER_JOB_RUN_DETAILS.

**See Also:**

"ALL_SCHEDULER_JOB_RUN_DETAILS"

## DBA_SCHEDULER_JOBS

DBA_SCHEDULER_JOBS displays information about all Scheduler jobs in the database. Its columns are the same as those in ALL_SCHEDULER_JOBS.

**See Also:**

"ALL_SCHEDULER_JOBS"

## DBA_SCHEDULER_NOTIFICATIONS

DBA_SCHEDULER_NOTIFICATIONS displays information about the E-mail notifications for all jobs in the database. Its columns are the same as those in ALL_SCHEDULER_NOTIFICATIONS.

**See Also:**

"ALL_SCHEDULER_NOTIFICATIONS"

## DBA_SCHEDULER_PROGRAM_ARGS

DBA_SCHEDULER_PROGRAM_ARGS displays information about the arguments of all Scheduler programs in the database. Its columns are the same as those in ALL_SCHEDULER_PROGRAM_ARGS.

**See Also:**

"ALL_SCHEDULER_PROGRAM_ARGS"

## DBA_SCHEDULER_PROGRAMS

DBA_SCHEDULER_PROGRAMS displays information about all Scheduler programs in the database. Its columns are the same as those in ALL_SCHEDULER_PROGRAMS.

**See Also:**

"ALL_SCHEDULER_PROGRAMS"

## DBA_SCHEDULER_REMOTE_DATABASES

DBA_SCHEDULER_REMOTE_DATABASES displays information about all remote databases that have been registered as sources and destinations for remote database jobs. Its columns are the same as those in ALL_SCHEDULER_REMOTE_DATABASES.

**See Also:**

"ALL_SCHEDULER_REMOTE_DATABASES"

## DBA_SCHEDULER_REMOTE_JOBSTATE

DBA_SCHEDULER_REMOTE_JOBSTATE displays information about the state of all jobs at remote databases. Its columns are the same as those in ALL_SCHEDULER_REMOTE_JOBSTATE.

**See Also:**

"ALL_SCHEDULER_REMOTE_JOBSTATE"

## DBA_SCHEDULER_RUNNING_CHAINS

DBA_SCHEDULER_RUNNING_CHAINS displays information about the chain steps of all running chains in the database. Its columns are the same as those in ALL_SCHEDULER_RUNNING_CHAINS.

**See Also:**

"ALL_SCHEDULER_RUNNING_CHAINS"

## DBA_SCHEDULER_RUNNING_JOBS

DBA_SCHEDULER_RUNNING_JOBS displays information about all running Scheduler jobs in the database. Its columns are the same as those in ALL_SCHEDULER_RUNNING_JOBS.

**See Also:**

"ALL_SCHEDULER_RUNNING_JOBS"

## DBA_SCHEDULER_SCHEDULES

DBA_SCHEDULER_SCHEDULES displays information about all Scheduler schedules in the database. Its columns are the same as those in ALL_SCHEDULER_SCHEDULES.

**See Also:**

"ALL_SCHEDULER_SCHEDULES"

## DBA_SCHEDULER_WINDOW_DETAILS

DBA_SCHEDULER_WINDOW_DETAILS displays log details for all Scheduler windows in the database. Its columns are the same as those in ALL_SCHEDULER_WINDOW_DETAILS.

**See Also:**

"ALL_SCHEDULER_WINDOW_DETAILS"

## DBA_SCHEDULER_WINDOW_GROUPS

DBA_SCHEDULER_WINDOW_GROUPS displays information about all Scheduler window groups in the database. Its columns are the same as those in ALL_SCHEDULER_WINDOW_GROUPS.

**See Also:**

"ALL_SCHEDULER_WINDOW_GROUPS"

## DBA_SCHEDULER_WINDOW_LOG

DBA_SCHEDULER_WINDOW_LOG displays log information for all Scheduler windows in the database. Its columns are the same as those in ALL_SCHEDULER_WINDOW_LOG.

**See Also:**

"ALL_SCHEDULER_WINDOW_LOG"

## DBA_SCHEDULER_WINDOWS

DBA_SCHEDULER_WINDOWS displays information about all Scheduler windows in the database. Its columns are the same as those in ALL_SCHEDULER_WINDOWS.

**See Also:**

"ALL_SCHEDULER_WINDOWS"

## DBA_SCHEDULER_WINGROUP_MEMBERS

DBA_SCHEDULER_WINGROUP_MEMBERS displays the members of all Scheduler window groups in the database. Its columns are the same as those in ALL_SCHEDULER_WINGROUP_MEMBERS.

**See Also:**

"ALL_SCHEDULER_WINGROUP_MEMBERS"

# DBA_SEC_RELEVANT_COLS

DBA_SEC_RELEVANT_COLS describes the security relevant columns of all security policies in the database. Its columns are the same as those in ALL_SEC_RELEVANT_COLS.

**See Also:**

"ALL_SEC_RELEVANT_COLS"

# DBA_SECONDARY_OBJECTS

DBA_SECONDARY_OBJECTS provides information about all secondary objects that are associated with domain indexes in the database. This view is only relevant in the context of domain indexes. Its columns are the same as those in "ALL_SECONDARY_OBJECTS".

# DBA_SEGMENTS

DBA_SEGMENTS describes the storage allocated for all segments in the database.

### Related View

USER_SEGMENTS describes the storage allocated for the segments owned by the current user's objects. This view does not display the OWNER, HEADER_FILE, HEADER_BLOCK, or RELATIVE_FNO columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Username of the segment owner |
| SEGMENT_NAME | VARCHAR2(128) | | Name, if any, of the segment |
| PARTITION_NAME | VARCHAR2(128) | | Object Partition Name (Set to NULL for nonpartitioned objects) |
| SEGMENT_TYPE | VARCHAR2(18) | | Type of segment:<br>• NESTED TABLE<br>• TABLE<br>• TABLE PARTITION<br>• CLUSTER<br>• LOBINDEX<br>• INDEX<br>• INDEX PARTITION<br>• LOBSEGMENT<br>• TABLE SUBPARTITION<br>• INDEX SUBPARTITION<br>• LOB PARTITION<br>• LOB SUBPARTITION<br>• ROLLBACK |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • TYPE2 UNDO<br>• DEFERRED ROLLBACK<br>• TEMPORARY<br>• CACHE<br>• SPACE HEADER<br>• UNDEFINED |
| SEGMENT_SUBTYPE | VARCHAR2(10) | | Subtype of LOB segment: SECUREFILE, ASSM, MSSM, and NULL |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the segment |
| HEADER_FILE | NUMBER | | ID of the file containing the segment header |
| HEADER_BLOCK | NUMBER | | ID of the block containing the segment header |
| BYTES | NUMBER | | Size, in bytes, of the segment |
| BLOCKS | NUMBER | | Size, in Oracle blocks, of the segment |
| EXTENTS | NUMBER | | Number of extents allocated to the segment |
| INITIAL_EXTENT | NUMBER | | Size in bytes requested for the initial extent of the segment at create time. (Oracle rounds the extent size to multiples of 5 blocks if the requested size is greater than 5 blocks.) |
| NEXT_EXTENT | NUMBER | | Size in bytes of the next extent to be allocated to the segment |
| MIN_EXTENTS | NUMBER | | Minimum number of extents allowed in the segment |
| MAX_EXTENTS | NUMBER | | Maximum number of extents allowed in the segment |
| MAX_SIZE | NUMBER | | Maximum number of blocks allowed in the segment |
| RETENTION | VARCHAR2(7) | | Retention option for SECUREFILE segment |
| MINRETENTION | NUMBER | | Minimum retention duration for SECUREFILE segment |
| PCT_INCREASE | NUMBER | | Percent by which to increase the size of the next extent to be allocated |
| FREELISTS | NUMBER | | Number of process freelists allocated to this segment |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated to this segment |
| RELATIVE_FNO | NUMBER | | Relative file number of the segment header |
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool to be used for segment blocks:<br>• DEFAULT<br>• KEEP<br>• RECYCLE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FLASH_CACHE | VARCHAR2(7) | | Database Smart Flash Cache hint to be used for segment blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | | Cell flash cache hint to be used for segment blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is enabled (ENABLED) or disabled (DISABLED) for this segment |
| INMEMORY_PRIORITY[1] | VARCHAR2(8) | | Indicates the priority for In-Memory Column Store (IM column store) population:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| INMEMORY_DISTRIBUTE[1] | VARCHAR2(15) | | Indicates how the IM column store is distributed in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| INMEMORY_DUPLICATE[1] | VARCHAR2(13) | | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |
| INMEMORY_COMPRESSION[1] | VARCHAR2(17) | | Indicates the compression level for the IM column store:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ]<br>• NULL |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2)

**See Also:**

"USER_SEGMENTS"

# DBA_SEGMENTS_OLD

DBA_SEGMENTS_OLD lists information about storage allocated for all database segments.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Username of the segment owner |
| SEGMENT_NAME | VARCHAR2(128) | | Name, if any, of the segment |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition |
| SEGMENT_TYPE | VARCHAR2(18) | | Type of segment: INDEX PARTITION, TABLE PARTITION, TABLE, CLUSTER, INDEX, ROLLBACK, DEFERRED ROLLBACK, TEMPORARY, CACHE, LOBSEGMENT and LOBINDEX |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the segment |
| HEADER_FILE | NUMBER | | ID of the file containing the segment header |
| HEADER_BLOCK | NUMBER | | ID of the block containing the segment header |
| BYTES | NUMBER | | Size, in bytes, of the segment |
| BLOCKS | NUMBER | | Size, in Oracle blocks, of the segment |
| EXTENTS | NUMBER | | Number of extents allocated to the segment |
| INITIAL_EXTENT | NUMBER | | Size in bytes requested for the initial extent of the segment at create time. (Oracle rounds the extent size to multiples of 5 blocks if the requested size is greater than 5 blocks.) |
| NEXT_EXTENT | NUMBER | | Size in bytes of the next extent to be allocated to the segment |
| MIN_EXTENTS | NUMBER | | Minimum number of extents allowed in the segment |
| MAX_EXTENTS | NUMBER | | Maximum number of extents allowed in the segment |
| PCT_INCREASE | NUMBER | | Percent by which to increase the size of the next extent to be allocated |
| FREELISTS | NUMBER | | Number of process freelists allocated to the segment |
| FREELIST_GROUPS | NUMBER | | Number of freelist groups allocated to this segment |
| RELATIVE_FNO | NUMBER | | Relative file number of the segment header |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BUFFER_POOL | VARCHAR2(7) | | Buffer pool for the object |

# DBA_SENSITIVE_COLUMN_TYPES

DBA_SENSITIVE_COLUMN_TYPES describes sensitive column types in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | | The name of the sensitive column type |
| USER_COMMENT | VARCHAR2(4000) | | User comment on the sensitive column type |
| SOURCE_NAME | VARCHAR2(128) | | The name of the discovery source for the sensitive column type |
| SOURCE_TYPE | VARCHAR2(3) | | The type of the discovery source:<br>• ADM: import from ADM<br>• DB: added within the database |

**See Also:**

*Oracle Database Security Guide* for more information about transparent sensitive data protection

# DBA_SENSITIVE_DATA

DBA_SENSITIVE_DATA describes the sensitive columns in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SENSITIVE# | NUMBER | NOT NULL | Dictionary ID for the sensitive data |
| SCHEMA_NAME | VARCHAR2(128) | NOT NULL | The schema containing the sensitive data |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | The table containing the sensitive data |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | The name of the column identified as sensitive |
| SENSITIVE_TYPE | VARCHAR2(128) | | The sensitive column type of the data |
| SOURCE_NAME | VARCHAR2(128) | | The name of the discovery source for the sensitive data |
| USER_COMMENT | VARCHAR2(4000) | | User comment on the sensitive data |
| TS | TIMESTAMP(6) | | The time when the data was identified as sensitive in the database |

**See Also:**

*Oracle Database Security Guide* for more information about transparent sensitive data protection

## DBA_SEQUENCES

DBA_SEQUENCES describes all sequences in the database. Its columns are the same as those in "ALL_SEQUENCES".

## DBA_SERVER_REGISTRY

DBA_SERVER_REGISTRY displays information about all server components in the database that are loaded into the component registry.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMP_ID | VARCHAR2(30) | NOT NULL | Component identifier |
| COMP_NAME | VARCHAR2(255) | | Component name |
| VERSION | VARCHAR2(30) | | Component version loaded |
| STATUS | VARCHAR2(11) | | Component status:<br>• INVALID<br>• VALID<br>• LOADING<br>• LOADED<br>• UPGRADING<br>• UPGRADED<br>• DOWNGRADING<br>• DOWNGRADED<br>• REMOVING<br>• REMOVED |
| MODIFIED | VARCHAR2(20) | | Time when the component was last modified |
| CONTROL | VARCHAR2(128) | NOT NULL | User that created the component entry |
| SCHEMA | VARCHAR2(128) | NOT NULL | User that contains the objects for the component |
| PROCEDURE | VARCHAR2(61) | | Validation procedure |
| STARTUP | VARCHAR2(8) | | Indicates whether the component requires a startup after the upgrade (REQUIRED) or not |
| PARENT_ID | VARCHAR2(30) | | Parent component identifier |
| OTHER_SCHEMAS | VARCHAR2(4000) | | A list of ancillary schema names associated with the component |

## DBA_SERVICES

DBA_SERVICES displays all services in the database. The view excludes rows marked for deletion. Its columns are the same as those in ALL_SERVICES.

**See Also:**

"ALL_SERVICES"

## DBA_SOURCE

DBA_SOURCE describes the text source of all stored objects in the database. Its columns are the same as those in "ALL_SOURCE".

## DBA_SOURCE_AE

DBA_SOURCE_AE describes the text source of all stored objects (across all editions) in the database. Its columns are the same as those in ALL_SOURCE_AE.

**See Also:**

"ALL_SOURCE_AE"

## DBA_SQL_MANAGEMENT_CONFIG

DBA_SQL_MANAGEMENT_CONFIG displays the configuration parameters of the SQL management base. You must have the DBA role in order to change the configuration parameter values.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETER_NAME | VARCHAR2(128) | NOT NULL | Name of the configuration parameter:<br>• SPACE_BUDGET_PERCENT<br>• PLAN_RETENTION_WEEKS |
| PARAMETER_VALUE | NUMBER | NOT NULL | Value of the configuration parameter |
| LAST_MODIFIED | TIMESTAMP(6) | | Time the parameter value was last updated |
| MODIFIED_BY | VARCHAR2(128) | | User who last updated the parameter value |

## DBA_SQL_PATCHES

DBA_SQL_PATCHES displays the set of SQL patches.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | NOT NULL | Name of the SQL patch |
| CATEGORY | VARCHAR2(128) | NOT NULL | Category of the SQL patch |
| SIGNATURE | NUMBER | NOT NULL | Unique identifier generated from normalized SQL text |
| SQL_TEXT | CLOB | NOT NULL | Un-normalized SQL text |
| CREATED | TIMESTAMP(6) | NOT NULL | Timestamp when the SQL patch was created |
| LAST_MODIFIED | TIMESTAMP(6) | | Timestamp when the SQL patch was last modified |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DESCRIPTION | VARCHAR2(500) | | Text description provided for the SQL patch |
| STATUS | VARCHAR2(8) | | Status of the SQL patch:<br>• ENABLED<br>• DISABLED |
| FORCE_MATCHING | VARCHAR2(3) | | Indicates whether the signature is force matching (YES) or exact matching (NO) |
| TASK_ID | NUMBER | | Advisor task ID that generated the SQL patch |
| TASK_EXEC_NAME | VARCHAR2(128) | | Advisor execution name for the SQL patch |
| TASK_OBJ_ID | NUMBER | | Advisor object ID for the SQL patch |
| TASK_FND_ID | NUMBER | | Advisor finding ID for the SQL patch |
| TASK_REC_ID | NUMBER | | Advisor recommendation ID for the SQL patch |

# DBA_SQL_PLAN_BASELINES

DBA_SQL_PLAN_BASELINES displays information about the SQL plan baselines currently created for specific SQL statements.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SIGNATURE | NUMBER | NOT NULL | Unique SQL identifier generated from normalized SQL text |
| SQL_HANDLE | VARCHAR2(128) | NOT NULL | Unique SQL identifier in string form as a search key |
| SQL_TEXT | CLOB | NOT NULL | Un-normalized SQL text |
| PLAN_NAME | VARCHAR2(128) | NOT NULL | Unique plan identifier in string form as a search key |
| CREATOR | VARCHAR2(128) | | User who created the plan baseline |
| ORIGIN | VARCHAR2(14) | | How the plan baseline was created:<br>• MANUAL-LOAD<br>• AUTO-CAPTURE<br>• MANUAL-SQLTUNE<br>• AUTO-SQLTUNE |
| PARSING_SCHEMA_NAME | VARCHAR2(128) | | Name of the parsing schema |
| DESCRIPTION | VARCHAR2(500) | | Text description provided for the plan baseline |
| VERSION | VARCHAR2(64) | | Database version at the time of plan baseline creation |
| CREATED | TIMESTAMP(6) | NOT NULL | Timestamp when the plan baseline was created |
| LAST_MODIFIED | TIMESTAMP(6) | | Timestamp when the plan baseline was last modified |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LAST_EXECUTED | TIMESTAMP(6) | | Timestamp when the plan baseline was last executed |
| LAST_VERIFIED | TIMESTAMP(6) | | Timestamp when the plan baseline was last verified |
| ENABLED | VARCHAR2(3) | | Indicates whether the plan baseline is enabled (YES) or disabled (NO) |
| ACCEPTED | VARCHAR2(3) | | Indicates whether the plan baseline is accepted (YES) or not (NO) |
| FIXED | VARCHAR2(3) | | Indicates whether the plan baseline is fixed (YES) or not (NO) |
| REPRODUCED | VARCHAR2(3) | | Indicates whether the optimizer was able to reproduce the plan (YES) or not (NO). The value of this column is set to YES when a plan is initially added to the plan baseline. |
| AUTOPURGE | VARCHAR2(3) | | Indicates whether the plan baseline is auto-purged (YES) or not (NO) |
| ADAPTIVE | VARCHAR2(3) | | Indicates whether a plan that is automatically captured by SQL plan management is marked adaptive or not.<br>When a new adaptive plan is found for a SQL statement that has an existing SQL plan baseline, that new plan will be added to the SQL plan baseline as an unaccepted plan, and the ADAPTIVE column will be marked YES. When this new plan is verified (either manually or via the auto evolve task), the plan will be test executed and the final plan determined at execution will become an accepted plan if its performance is better than the existing plan baseline. At this point, the value of the ADAPTIVE column is set to NO since the plan is no longer adaptive, but resolved. |
| OPTIMIZER_COST | NUMBER | | Optimizer cost at the time the plan baseline was created |
| MODULE | VARCHAR2(64) | | Application module name |
| ACTION | VARCHAR2(64) | | Application action |
| EXECUTIONS | NUMBER | | Number of executions at the time the plan baseline was created |
| ELAPSED_TIME | NUMBER | | Total elapsed time at the time the plan baseline was created |
| CPU_TIME | NUMBER | | Total CPU time at the time the plan baseline was created |
| BUFFER_GETS | NUMBER | | Total buffer gets at the time the plan baseline was created |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DISK_READS | NUMBER | | Total disk reads at the time the plan baseline was created |
| DIRECT_WRITES | NUMBER | | Total direct writes at the time the plan baseline was created |
| ROWS_PROCESSED | NUMBER | | Total rows processed at the time the plan baseline was created |
| FETCHES | NUMBER | | Total number of fetches at the time the plan baseline was created |
| END_OF_FETCH_COUNT | NUMBER | | Total number of full fetches at the time the plan baseline was created |

**See Also:**

- *Oracle Database SQL Tuning Guide* for more information about SQL plan baselines
- The DBMS_SQLTUNE package in *Oracle Database PL/SQL Packages and Types Reference*

# DBA_SQL_PLAN_DIR_OBJECTS

DBA_SQL_PLAN_DIR_OBJECTS displays the objects created in the SQL plan directive.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DIRECTIVE_ID | NUMBER | | The identifier of the SQL plan directive |
| OWNER | VARCHAR2(128) | | The username of the owner of the object in the SQL plan directive |
| OBJECT_NAME | VARCHAR2(128) | | The name of the object in the SQL plan directive |
| SUBOBJECT_NAME | VARCHAR2(128) | | The name of the subobject (for example, column) in the SQL plan directive |
| OBJECT_TYPE | VARCHAR2(6) | | The type of the subobject in the SQL plan directive |
| NOTES | XMLTYPE | | Other notes about the object |

**See Also:**

- "DBA_SQL_PLAN_DIRECTIVES"
- *Oracle Database SQL Tuning Guide* for more information about SQL plan directives

# DBA_SQL_PLAN_DIRECTIVES

DBA_SQL_PLAN_DIRECTIVES displays information about the SQL plan directives in the system.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DIRECTIVE_ID | NUMBER | NOT NULL | The identifier of the SQL plan directive |
| TYPE | VARCHAR2(16) | | The type of the SQL plan directive |
| ENABLED | VARCHAR2(3) | | Indicates whether the SQL plan directive is enabled. Possible values:<br>• YES: The SQL plan directive is enabled.<br>• NO: The SQL plan directive is not enabled. |
| STATE | VARCHAR2(10) | | The state of the SQL plan directive. Possible values include:<br>• SUPERSEDED: This value indicates that the corresponding column or groups have an extension or histogram, or that another SQL plan directive exists that can be used for the directive.<br>• USABLE: This value indicates that the SQL plan directive is usable for the optimizer. |
| AUTO_DROP | VARCHAR2(3) | | If YES, the SQL plan directive gets dropped when unused beyond SPD_RETENTION_WEEKS |
| REASON | VARCHAR2(36) | | The reason for creating the SQL plan directive |
| CREATED | TIMESTAMP(6) | | The creation timestamp of the SQL plan directive |
| LAST_MODIFIED | TIMESTAMP(6) | | The timestamp of most recent modification of the SQL plan directive |
| LAST_USED | TIMESTAMP(9) | | The timestamp of most recent usage of the SQL plan directive |
| NOTES[1] | XMLTYPE | | Extra information about the SQL plan directive |

1 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "DBA_SQL_PLAN_DIR_OBJECTS"
- *Oracle Database SQL Tuning Guide* for more information about SQL plan directives

# DBA_SQL_PROFILES

DBA_SQL_PROFILES displays information about SQL profiles currently created for specific SQL statements.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | NOT NULL | Name of the SQL profile |
| CATEGORY | VARCHAR2(128) | NOT NULL | Category of the SQL profile |
| SIGNATURE | NUMBER | NOT NULL | Unique identifier generated from normalized SQL text |
| SQL_TEXT | CLOB | NOT NULL | Un-normalized SQL text |
| CREATED | TIMESTAMP(6) | NOT NULL | Timestamp when the SQL profile was created |
| LAST_MODIFIED | TIMESTAMP(6) | | Timestamp when the SQL profile was last modified |
| DESCRIPTION | VARCHAR2(500) | | Text description provided for the SQL profile |
| TYPE | VARCHAR2(7) | | Type of the SQL profile (how it was created):<br>• MANUAL<br>• AUTO |
| STATUS | VARCHAR2(8) | | Status of the SQL profile:<br>• ENABLED<br>• DISABLED<br>• VOID |
| FORCE_MATCHING | VARCHAR2(3) | | If YES, this causes SQL Profiles to target all SQL statements which have the same text after normalizing all literal values to bind variables. If a combination of literal values and bind variables is used in the same SQL text, then no transformation occurs. This is analogous to the matching algorithm use by the FORCE option of the CURSOR_SHARING parameter.<br>If NO, literals are not transformed. This is analogous to the matching algorithm used by the EXACT option of the CURSOR_SHARING parameter. |
| TASK_ID | NUMBER | | Advisor task ID that generated the SQL profile |
| TASK_EXEC_NAME | VARCHAR2(128) | | Advisor execution name for the SQL profile |
| TASK_OBJ_ID | NUMBER | | Advisor object ID for the SQL profile |
| TASK_FND_ID | NUMBER | | Advisor finding ID for the SQL profile |
| TASK_REC_ID | NUMBER | | Advisor recommendation ID for the SQL profile |
| TASK_CON_DBID | NUMBER | | Database ID for the PDB tuning task generating the SQL profile |

**See Also:**

The DBMS_SQLTUNE package in *Oracle Database PL/SQL Packages and Types Reference*

## DBA_SQL_TRANSLATION_PROFILES

DBA_SQL_TRANSLATION_PROFILES describes all SQL translation profiles in the database. Its columns are the same as those in ALL_SQL_TRANSLATION_PROFILES.

**See Also:**

"ALL_SQL_TRANSLATION_PROFILES"

## DBA_SQL_TRANSLATIONS

DBA_SQL_TRANSLATIONS describes all SQL translations in the database. Its columns are the same as those in ALL_SQL_TRANSLATIONS.

**See Also:**

"ALL_SQL_TRANSLATIONS"

## DBA_SQLJ_TYPE_ATTRS

DBA_SQLJ_TYPE_ATTRS describes the attributes of all SQLJ object types in the database. Its columns are the same as those in ALL_SQLJ_TYPE_ATTRS.

**See Also:**

"ALL_SQLJ_TYPE_ATTRS"

## DBA_SQLJ_TYPE_METHODS

DBA_SQLJ_TYPE_METHODS describes the methods of all SQLJ object types in the database. Its columns are the same as those in ALL_SQLJ_TYPE_METHODS.

**See Also:**

"ALL_SQLJ_TYPE_METHODS"

## DBA_SQLJ_TYPES

DBA_SQLJ_TYPES describes all SQLJ object types in the database. Its columns are the same as those in ALL_SQLJ_TYPES.

**See Also:**

"ALL_SQLJ_TYPES"

# DBA_SQLSET

DBA_SQLSET displays information about all SQL tuning sets in the database. Its columns are the same as those in "ALL_SQLSET".

# DBA_SQLSET_BINDS

DBA_SQLSET_BINDS displays the bind values associated with all SQL tuning sets in the database. Its columns are the same as those in "ALL_SQLSET_BINDS".

# DBA_SQLSET_PLANS

DBA_SQLSET_PLANS describes captured plans in the SQL tuning sets in the database. Its columns are the same as those in "ALL_SQLSET_PLANS".

# DBA_SQLSET_REFERENCES

DBA_SQLSET_REFERENCES describes whether or not all SQL tuning sets in the database are active. A SQL tuning set cannot be dropped if it is referenced. Its columns are the same as those in "ALL_SQLSET_REFERENCES".

# DBA_SQLSET_STATEMENTS

DBA_SQLSET_STATEMENTS displays information about the SQL statements, along with their statistics, that form all SQL tuning sets in the database. Its columns, except for PARSING_SCHEMA_ID, are the same as those in "ALL_SQLSET_STATEMENTS".

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SQLSET_NAME | VARCHAR2(128) | NOT NULL | Name of the SQL tuning set for the statement |
| SQLSET_OWNER | VARCHAR2(128) | | User name of the SQL tuning set owner |
| SQLSET_ID | NUMBER | NOT NULL | ID of the SQL tuning set for the statement |
| CON_DBID | NUMBER | NOT NULL | The database ID of the PDB |
| SQL_ID | VARCHAR2(13) | NOT NULL | SQL identifier of the parent cursor in the library cache |
| FORCE_MATCHING_SIGNATURE | NUMBER | NOT NULL | The signature used when the CURSOR_SHARING parameter is set to FORCE |
| SQL_TEXT | CLOB | | Full text for the SQL statement exposed as a CLOB column. |
| PARSING_SCHEMA_NAME | VARCHAR2(128) | | Name of the user in whose schema the statement was parsed |
| PARSING_SCHEMA_ID | NUMBER | | ID of the schema in which the statement was parsed |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Hash value for the plan corresponding to statistics in this record |
| BIND_DATA | RAW(2000) | | Bind data |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BINDS_CAPTURED | CHAR(1) | | Binds captured |
| MODULE | VARCHAR2(64) | | Contains the name of the module that was executing at the time that the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_MODULE |
| ACTION | VARCHAR2(64) | | Contains the name of the action that was executing at the time that the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_ACTION |
| ELAPSED_TIME | NUMBER | | Elapsed time (in microseconds) used by this cursor for parsing, executing, and fetching |
| CPU_TIME | NUMBER | | CPU time (in microseconds) used by this cursor for parsing, executing, and fetching |
| BUFFER_GETS | NUMBER | | Number of buffer gets for this child cursor |
| DISK_READS | NUMBER | | Number of disk reads for this child cursor |
| DIRECT_WRITES | NUMBER | | Number of direct writes for this child cursor |
| ROWS_PROCESSED | NUMBER | | Total number of rows that the parsed SQL statement returns |
| FETCHES | NUMBER | | Number of fetches associated with the SQL statement |
| EXECUTIONS | NUMBER | | Number of executions that took place on this object since it was brought into the library cache |
| END_OF_FETCH_COUNT | NUMBER | | Number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic in not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less than, or equal to, the value of the EXECUTIONS column. |
| OPTIMIZER_COST | NUMBER | | Cost of this query, given by the optimizer |
| OPTIMIZER_ENV | RAW(2000) | | Optimizer environment |
| PRIORITY | NUMBER | | User-defined priority |
| COMMAND_TYPE | NUMBER | | Oracle command type definition |
| FIRST_LOAD_TIME | VARCHAR2(19) | | Timestamp of the parent creation time |
| STAT_PERIOD | NUMBER | | Period of time (in seconds) during which the statistics of the SQL statement were collected |
| ACTIVE_STAT_PERIOD | NUMBER | | Effective period of time (in seconds) during which the SQL statement was active |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OTHER | CLOB | | Client data, specified by the user, for this statement |
| PLAN_TIMESTAMP | DATE | | Timestamp for the plan corresponding to the statistics in this record |
| SQL_SEQ | NUMBER | NOT NULL | SQL sequence |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_MODULE procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_ACTION procedure

# DBA_SQLTUNE_BINDS

DBA_SQLTUNE_BINDS displays the bind values associated with all tuned SQL statements in the database.

### Related View

USER_SQLTUNE_BINDS displays the bind values associated with the tuned SQL statements owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER(38) | NOT NULL | Tuning task identifier |
| OBJECT_ID | NUMBER(38) | NOT NULL | Advisor framework object identifier |
| POSITION | NUMBER(38) | NOT NULL | Bind position |
| VALUE | ANYDATA | | Bind value. This column is NULL for PL/SQL bind types. |

**See Also:**

"USER_SQLTUNE_BINDS"

# DBA_SQLTUNE_PLANS

DBA_SQLTUNE_PLANS displays information about the execution plans generated for all SQL statements in the database during a SQL tuning session.

### Related View

USER_SQLTUNE_PLANS displays information about the execution plans generated for the SQL statements owned by the current user during a SQL tuning session.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER(38) | NOT NULL | Advisor task ID |
| EXECUTION_NAME | VARCHAR2(128) | NOT NULL | Advisor task execution |
| OBJECT_ID | NUMBER(38) | NOT NULL | Advisor object ID |
| ATTRIBUTE | VARCHAR2(34) | | Text string identifying the type of the execution plan:<br>• `Original` - Original plan of the query<br>• `Original with adjusted cost` - Same as `Original` but with adjusted cost<br>• `Using SQL profile` - Plan with SQL profile applied<br>• `Using new indices` - Plan with indexes applied |
| STATEMENT_ID | VARCHAR2(30) | | Optional statement identifier specified in the `EXPLAIN PLAN` statement |
| PLAN_HASH_VALUE | NUMBER | NOT NULL | Numerical representation of the execution plan |
| PLAN_ID | NUMBER | | Plan identifier |
| TIMESTAMP | DATE | | Date and time when the `EXPLAIN PLAN` statement was issued |
| REMARKS | VARCHAR2(4000) | | Place for comments that can be added to the steps of the execution plan |
| OPERATION | VARCHAR2(30) | | Name of the operation performed at this step |
| OPTIONS | VARCHAR2(255) | | Options used for the operation performed at this step |
| OBJECT_NODE | VARCHAR2(128) | | Name of the database link used to reference the object |
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object |
| OBJECT_ALIAS | VARCHAR2(261) | | Object alias |
| OBJECT_INSTANCE | NUMBER(38) | | Numbered position of the object name in the original SQL statement |
| OBJECT_TYPE | VARCHAR2(30) | | Descriptive modifier that further describes the type of object |
| OPTIMIZER | VARCHAR2(255) | | Current mode of the optimizer |
| SEARCH_COLUMNS | NUMBER | | Number of index columns with start and stop keys (that is, the number of columns with matching predicates) |
| ID | NUMBER(38) | NOT NULL | Identification number for this step in the execution plan |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARENT_ID | NUMBER(38) | | ID of the next step that operates on the results of this step |
| DEPTH | NUMBER(38) | | Depth |
| POSITION | NUMBER(38) | | Order of processing for steps with the same parent ID |
| COST | NUMBER(38) | | Cost of the current operation estimated by the cost-based optimizer (CBO) |
| CARDINALITY | NUMBER(38) | | Number of rows returned by the current operation (estimated by the CBO) |
| BYTES | NUMBER(38) | | Number of bytes returned by the current operation |
| OTHER_TAG | VARCHAR2(255) | | Describes the function of the SQL text in the OTHER column. Values for OTHER_TAG are:<br>• SERIAL - SQL is the text of a locally-executed, serial query plan. Currently, SQL is not loaded in OTHER for this case.<br>• SERIAL_FROM_REMOTE - SQL text shown in the OTHER column will be executed at a remote site<br>• PARALLEL_COMBINED_WITH_PARENT - Parent of this operation is a DFO that performs both operations in the parallel execution plan<br>• PARALLEL_COMBINED_WITH_CHILD - Child of this operation is a DFO that performs both operations in the parallel execution plan.<br>• PARALLEL_TO_SERIAL - SQL text shown in the OTHER column is the top-level of the parallel plan.<br>• PARALLEL_TO_PARALLEL - SQL text shown in the OTHER column is executed and output in parallel<br>• PARALLEL_FROM_SERIAL - Operation consumes data from a serial operation and outputs it in parallel |
| PARTITION_START | VARCHAR2(255) | | Start partition of a range of accessed partitions |
| PARTITION_STOP | VARCHAR2(255) | | Stop partition of a range of accessed partitions |
| PARTITION_ID | NUMBER(38) | | Step that has computed the pair of values of the PARTITION_START and PARTITION_STOP columns |
| OTHER | LONG | | Information about parallel execution servers and parallel queries |
| DISTRIBUTION | VARCHAR2(30) | | Distribution method |
| CPU_COST | NUMBER(38) | | User-defined CPU cost |
| IO_COST | NUMBER(38) | | User-defined I/O cost |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TEMP_SPACE | NUMBER(38) | | Temporary space usage of the operation (sort or hash-join) as estimated by the CBO |
| ACCESS_PREDICATES | VARCHAR2(4000) | | Predicates used to locate rows in an access structure. For example, start or stop predicates for an index range scan. |
| FILTER_PREDICATES | VARCHAR2(4000) | | Predicates used to filter rows before producing them |
| PROJECTION | VARCHAR2(4000) | | Expressions produced by the operation |
| TIME | NUMBER(38) | | Elapsed time (in seconds) of the operation as estimated by the CBO |
| QBLOCK_NAME | VARCHAR2(128) | | Name of the query block |
| OTHER_XML | CLOB | | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML because it allows multiple pieces of information to be stored, including the following:<br>• Name of the schema against which the query was parsed<br>• Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan<br>• Indication of whether or not dynamic statistics were used to produce the plan<br>• The outline data, a set of optimizer hints that can be used to regenerate the same plan |

**See Also:**

"USER_SQLTUNE_PLANS"

# DBA_SQLTUNE_RATIONALE_PLAN

DBA_SQLTUNE_RATIONALE_PLAN displays the association between rationales and operations in the execution plan of all SQL statements in the database.

### Related View

USER_SQLTUNE_RATIONALE_PLAN displays the association between rationales and operations in the execution plan of the SQL statements owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER(38) | NOT NULL | Tuning task identifier |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXECUTION_NAME | VARCHAR2(128) | NOT NULL | The name of the task execution with which this entry (row) is associated |
| RATIONALE_ID | NUMBER(38) | NOT NULL | Rationale identifier |
| OBJECT_ID | NUMBER(38) | NOT NULL | Advisor framework object identifier |
| OPERATION_ID | NUMBER(38) | NOT NULL | Operation identifier |
| PLAN_ATTRIBUTE | VARCHAR2(34) | | Type of the execution plan:<br>• `Original` - Original plan of the query<br>• `Original with adjusted cost` - Same as `Original` but with adjusted cost<br>• `Using SQL profile` - Plan with SQL profile applied<br>• `Using new indices` - Plan with indexes applied |

**See Also:**

"USER_SQLTUNE_RATIONALE_PLAN"

# DBA_SQLTUNE_STATISTICS

DBA_SQLTUNE_STATISTICS displays statistics associated with all SQL statements in the database.

### Related View

USER_SQLTUNE_STATISTICS displays statistics associated with the SQL statements owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TASK_ID | NUMBER(38) | NOT NULL | Tuning task identifier |
| OBJECT_ID | NUMBER(38) | NOT NULL | Advisor framework object identifier |
| PARSING_SCHEMA_ID | NUMBER | | Schema under which the SQL is parsed |
| MODULE | VARCHAR2(64) | | Last application module recorded for the SQL |
| ACTION | VARCHAR2(64) | | Last application action recorded for the SQL |
| ELAPSED_TIME | NUMBER | | Elapsed time for the SQL statement |
| CPU_TIME | NUMBER | | CPU time for the SQL |
| BUFFER_GETS | NUMBER | | Number of buffer gets |
| DISK_READS | NUMBER | | Number of disk reads |
| DIRECT_WRITES | NUMBER | | Number of disk writes |
| ROWS_PROCESSED | NUMBER | | Number of rows processed by the SQL |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FETCHES | NUMBER | | Number of fetches |
| EXECUTIONS | NUMBER | | Number of executions |
| END_OF_FETCH_COUNT | NUMBER | | End of fetch count |
| OPTIMIZER_COST | NUMBER | | Optimizer cost for the SQL |
| OPTIMIZER_ENV | RAW(2000) | | Optimizer environment |
| COMMAND_TYPE | NUMBER | | Command type |

**See Also:**

"USER_SQLTUNE_STATISTICS"

# DBA_SR_GRP_STATUS

DBA_SR_GRP_STATUS provides information on the current refresh operations for the current synchronous refresh groups in the database. It has the same columns as the DBA_SR_GRP_STATUS_ALL view.

Refresh operations are controlled using the DBMS_SYNC_REFRESH package.

### Related View

USER_SR_GRP_STATUS provides information on the current refresh operations for the current synchronous refresh groups in the database which are owned by the current user. Its columns are the same as those in DBA_SR_GRP_STATUS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the refresh operation, which is the user who launched the operation |
| GROUP_ID | NUMBER | NOT NULL | Group ID of the synchronous refresh group |
| OPERATION | VARCHAR2(7) | | The phase of the refresh operation performed:<br>• PREPARE<br>• EXECUTE |
| STATUS | VARCHAR2(10) | | The status of the refresh operation:<br>• RUNNING<br>• NOT PROCESSED<br>• COMPLETE<br>• ERROR-SOFT<br>• ERROR-HARD<br>• ABORT<br>• PARTIAL |
| NUM_TBLS | NUMBER | NOT NULL | The number of tables in the synchronous refresh group |
| NUM_MVS | NUMBER | NOT NULL | The number of materialized views in the synchronous refresh group |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| BASE_TBLS_REFR_STATUS | VARCHAR2(13) | | Indicates the refresh status of base tables in the synchronous refresh group. The possible values are:<br>• NOT PROCESSED<br>• COMPLETE<br>• ABORT |
| NUM_MVS_COMPLETED | NUMBER | | The number of materialized views which have completed refresh in the synchronous refresh group |
| NUM_MVS_ABORTED | NUMBER | | The number of materialized views which have aborted refresh in the synchronous refresh group |
| ERROR_NUMBER | NUMBER | | Error number of the run (if any) |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message of the run (if any) |
| PREPARE_START_TIME | DATE | | Time that the PREPARE_REFRESH phase of the run started |
| PREPARE_END_TIME | DATE | | Time that the PREPARE_REFRESH phase of the run ended |
| EXECUTE_START_TIME | DATE | | Time that the EXECUTE_REFRESH phase of the run started |
| EXECUTE_END_TIME | DATE | | Time that the EXECUTE_REFRESH phase of the run ended |

**See Also:**

- "USER_SR_GRP_STATUS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SYNC_REFRESH package

# DBA_SR_GRP_STATUS_ALL

DBA_SR_GRP_STATUS_ALL provides information on the refresh operations on the synchronous refresh groups in the database. A refresh operation is also called a run, and it has two phases: PREPARE_REFRESH and EXECUTE_REFRESH. These phases are controlled using the DBMS_SYNC_REFRESH package.

Each row in this view provides information on a run of a group, identified by its GROUP_ID. The view contains information on the status of the objects of both current and past runs of both current and defunct synchronous refresh groups. Therefore, this view can be used to examine the history of synchronous refresh operations.

The current run of a group is the most recent run of a group; a current group is a currently valid group, which is capable of being refreshed. A group becomes defunct when it is unregistered for any reason, either explicitly by the user or implicitly as a side-effect when the user registers materialized views related to the materialized views in the group.

To view the status of refresh operations for the most recent runs of only the current groups, use the DBA_SR_GRP_STATUS view.

### Related View

USER_SR_GRP_STATUS_ALL provides information on the refresh operations on the synchronous refresh groups in the database which are owned by the current user. Its columns are the same as those in DBA_SR_GRP_STATUS_ALL.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the refresh operation, which is the user who launched the operation |
| GROUP_ID | NUMBER | NOT NULL | Group ID of the synchronous refresh group |
| OPERATION | VARCHAR2(7) | | The phase of the refresh operation performed:<br>• PREPARE<br>• EXECUTE |
| STATUS | VARCHAR2(10) | | The status of the refresh operation:<br>• RUNNING<br>• NOT PROCESSED<br>• COMPLETE<br>• ERROR-SOFT<br>• ERROR-HARD<br>• ABORT<br>• PARTIAL |
| CURRENT_RUN | VARCHAR2(1) | | Indicates whether the record is for the most recent refresh-operation on the group:<br>• Y - Yes<br>• N - No |
| CURRENT_GROUP | VARCHAR2(1) | | Indicates whether the record is for a current group:<br>• Y - Yes<br>• N - No |
| NUM_TBLS | NUMBER | NOT NULL | The number of tables in the synchronous refresh group |
| NUM_MVS | NUMBER | NOT NULL | The number of materialized views in the synchronous refresh group |
| BASE_TBLS_REFR_STATUS | VARCHAR2(13) | | Indicates the refresh status of base tables in the synchronous refresh group. The possible values are:<br>• NOT PROCESSED<br>• COMPLETE<br>• ABORT |
| NUM_MVS_COMPLETED | NUMBER | | The number of materialized views which have completed refresh in the synchronous refresh group |
| NUM_MVS_ABORTED | NUMBER | | The number of materialized views which have aborted refresh in the synchronous refresh group |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ERROR_NUMBER | NUMBER | | Error number of the run (if any) |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message of the run (if any) |
| PREPARE_START_TIME | DATE | | Time that the PREPARE_REFRESH phase of the run started |
| PREPARE_END_TIME | DATE | | Time that the PREPARE_REFRESH phase of the run ended |
| EXECUTE_START_TIME | DATE | | Time that the EXECUTE_REFRESH phase of the run started |
| EXECUTE_END_TIME | DATE | | Time that the EXECUTE_REFRESH phase of the run ended |

**See Also:**

- "USER_SR_GRP_STATUS_ALL"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SYNC_REFRESH package

# DBA_SR_OBJ

DBA_SR_OBJ provides information on the objects registered for synchronous refresh for current groups.

### Related View

USER_SR_OBJ provides information on the objects registered for synchronous refresh for current groups for the current user. Its columns are the same as those in DBA_SR_OBJ.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the synchronous refresh object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the synchronous refresh object |
| TYPE | VARCHAR2(5) | | Type of synchronous refresh object:<br>• MVIEW<br>• TABLE |
| GROUP_ID | NUMBER | NOT NULL | Group ID of the synchronous refresh group to which this object belongs |
| STAGING_LOG_NAME | VARCHAR2(128) | | Name of the staging log for tables. This column has a value of NULL for materialized views. |

**See Also:**

"USER_SR_OBJ"

# DBA_SR_OBJ_ALL

DBA_SR_OBJ_ALL provides information on the objects registered for synchronous refresh for current and defunct groups.

To see information on the objects registered for synchronous refresh for only the current groups, use the DBA_SR_OBJ view.

### Related View

USER_SR_OBJ_ALL provides information on the objects registered for synchronous refresh for current and defunct groups for the current user. Its columns are the same as those in DBA_SR_OBJ_ALL.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the synchronous refresh object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the synchronous refresh object |
| TYPE | VARCHAR2(5) | | Type of synchronous refresh object:<br>• MVIEW<br>• TABLE |
| GROUP_ID | NUMBER | NOT NULL | Group ID of the synchronous refresh group to which this object belongs |
| CURRENT_GROUP | VARCHAR2(1) | | Indicates whether the record is for a current group:<br>• Y - Yes<br>• N - No |
| STAGING_LOG_NAME | VARCHAR2(128) | | Name of the staging log for tables. This column has a value of NULL for materialized views. |

**See Also:**

"USER_SR_OBJ_ALL"

# DBA_SR_OBJ_STATUS

DBA_SR_OBJ_STATUS provides information on the status of objects registered for synchronous refresh for the current refresh operations for the current synchronous refresh groups in the database.

To see information on the status of objects registered for synchronous refresh, use the DBA_SR_OBJ_STATUS_ALL view.

### Related View

USER_SR_OBJ_STATUS provides information on the status of objects registered for synchronous refresh for the current refresh operations for the current synchronous refresh groups in the database which are owned by the current user. Its columns are the same as those in DBA_SR_OBJ_STATUS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the synchronous refresh object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the synchronous refresh object |
| TYPE | VARCHAR2(5) | | Type of synchronous refresh object:<br>• MVIEW<br>• TABLE |
| GROUP_ID | NUMBER | NOT NULL | Group ID of the synchronous refresh group to which this object belongs |
| STATUS | VARCHAR2(13) | | Status of the synchronous refresh object:<br>• NOT PROCESSED<br>• COMPLETE<br>• ABORT |
| ERROR_NUMBER | NUMBER | | Error number of the run (if any) |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message of the run (if any) |
| LAST_MODIFIED_TIME | DATE | | Last modification time of the synchronous refresh object |

**See Also:**

"USER_SR_OBJ_STATUS"

# DBA_SR_OBJ_STATUS_ALL

DBA_SR_OBJ_STATUS_ALL provides information on the status of objects registered for synchronous refresh.

The view contains information on the status of the objects of both the current and past runs of both current and defunct groups. Therefore, this view can be used to examine the history of synchronous refresh operations. The current run of a group is the most recent run of a group; a current group is a currently valid group, which is capable of being refreshed. A group becomes defunct when it is unregistered for any reason, either explicitly by the user or implicitly as a side-effect when the user registers materialized views related to the materialized views in the group.

To see information on the status of refresh operations for the most recent runs of only the current groups, use the DBA_SR_OBJ_STATUS view.

### Related View

USER_SR_OBJ_STATUS_ALL provides information on the status of objects registered for synchronous refresh in the database which are owned by the current user. Its columns are the same as those in DBA_SR_OBJ_STATUS_ALL.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the synchronous refresh object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the synchronous refresh object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE | VARCHAR2(5) | | Type of synchronous refresh object:<br>• MVIEW<br>• TABLE |
| GROUP_ID | NUMBER | NOT NULL | Group ID of the synchronous refresh group to which this object belongs |
| STATUS | VARCHAR2(13) | | Status of the synchronous refresh object:<br>• NOT PROCESSED<br>• COMPLETE<br>• ABORT |
| CURRENT_RUN | VARCHAR2(1) | | Indicates whether the record is for the most recent refresh operation on the group:<br>• Y - Yes<br>• N - No |
| CURRENT_GROUP | VARCHAR2(1) | | Indicates whether the record is for a current group:<br>• Y - Yes<br>• N - No |
| ERROR_NUMBER | NUMBER | | Error number of the run (if any) |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message of the run (if any) |
| LAST_MODIFIED_TIME | DATE | | Last modification time of the synchronous refresh object |

**See Also:**

"USER_SR_OBJ_STATUS_ALL"

# DBA_SR_PARTN_OPS

DBA_SR_PARTN_OPS provides information on the partition operations registered on the base tables of the materialized views registered for synchronous refresh. These rows last only as long as the registrations are active; that is, they disappear after EXECUTE_REFRESH or ABORT_REFRESH of the base table by the DBMS_SYNC_REFRESH package.

### Related View

USER_SR_PARTN_OPS provides information on the partition operations registered on the base tables of the materialized views registered for synchronous refresh belonging to the current user. Its columns are the same as those in DBA_SR_PARTN_OPS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the base table registered for synchronous refresh |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARTITION_OP | VARCHAR2(30) | NOT NULL | Type of partition operation:<br>• DROP<br>• EXCHANGE<br>• TRUNCATE |
| PARTITION_NAME | VARCHAR2(30) | NOT NULL | Name of the partition to be changed |
| OUTSIDE_TABLE_SCHEMA | VARCHAR2(30) | | Schema in which the outside table (for EXCHANGE PARTITION) was created |
| OUTSIDE_TABLE_NAME | VARCHAR2(30) | | Name of the outside table (for EXCHANGE PARTITION) |

**See Also:**

- "USER_SR_PARTN_OPS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SYNC_REFRESH package

# DBA_SR_STLOG_EXCEPTIONS

DBA_SR_STLOG_EXCEPTIONS provides information on the exceptions in the staging logs for the tables processed by DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG.

### Related View

USER_SR_STLOG_EXCEPTIONS provides information on the exceptions in the staging logs for the tables belonging to the current user processed by DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG. Its columns are the same as those in DBA_SR_STLOG_EXCEPTIONS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the base table registered for synchronous refresh |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the base table registered for synchronous refresh |
| STAGING_LOG_NAME | VARCHAR2(128) | NOT NULL | Name of the staging log for tables. This column has a value of NULL for materialized views. |
| BAD_ROWID | ROWID | NOT NULL | Row ID of the staging log row causing the exception for the synchronous refresh |
| ERROR_NUMBER | NUMBER | | Error number of the exception for the synchronous refresh |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message associated with the ERROR_NUMBER for the synchronous refresh |

**See Also:**

- "USER_SR_STLOG_EXCEPTIONS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SYNC_REFRESH package

# DBA_SR_STLOG_STATS

DBA_SR_STLOG_STATS provides information on the statistics in the staging logs for the tables processed by DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG.

These three statistics columns in the staging log are filled in PREPARE_STAGING_LOG:

- The number of inserts (NUM_INSERTS)
- The number of deletes (NUM_DELETES)
- The number of updates (NUM_UPDATES)

After the data in the staging logs of a synchronous refresh group have been processed by PREPARE_REFRESH and EXECUTE_REFRESH, the statistics columns for the tables in the group are cleared and appear as NULL.

### Related View

USER_SR_STLOG_STATS provides information on the statistics in the staging logs for the tables belonging to the current user processed by DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the base table registered for synchronous refresh |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table |
| STAGING_LOG_NAME | VARCHAR2(128) | NOT NULL | Name of the staging log for tables. NULL for materialized views |
| NUM_INSERTS | NUMBER | NOT NULL | The number of inserts in the staging log |
| NUM_DELETES | NUMBER | NOT NULL | The number of deletes in the staging log |
| NUM_UPDATES | NUMBER | NOT NULL | The number of updates in the staging log |
| PSL_MODE | VARCHAR2(33) | | The mode specified by the user in running DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG. Possible values:<br>• DELETE_TRUSTED<br>• DELETE_TRUSTED and UPDATE_TRUSTED<br>• ENFORCED<br>• INSERT_TRUSTED<br>• INSERT_TRUSTED and DELETE_TRUSTED<br>• TRUSTED<br>• UPDATE_TRUSTED<br>• UPDATE_TRUSTED and INSERT_TRUSTED |

**See Also:**

- "USER_SR_STLOG_STATS"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SYNC_REFRESH package

# DBA_SSCR_CAPTURE

DBA_SSCR_CAPTURE displays session state capture statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DB_NAME | VARCHAR2(4000) | | Database name of captured session |
| INST_NAME | VARCHAR2(4000) | | Instance name of captured session |
| INST_ID | NUMBER | | Instance ID of captured session |
| SESSION_ID | NUMBER | | Session ID of captured session |
| SESSION_SERIAL# | NUMBER | | Session serial number of captured session |
| USER_NAME | VARCHAR2(128) | NOT NULL | User name of captured session |
| SCHEMA_NAME | VARCHAR2(128) | NOT NULL | Schema name of captured session |
| SEQUENCE# | NUMBER | | Sequence number of captured session |
| CAPTURE_MODE | VARCHAR2(7) | | Mode of capture operation |
| CAPTURE_SCOPE | VARCHAR2(7) | | Scope of capture operation |
| CAPTURE_FORMAT | VARCHAR2(9) | | Format of capture files |
| CAPTURE_DIR | VARCHAR2(128) | | Directory object of capture files |
| CAPTURE_LOCATOR | RAW(64) | | Locator of master capture file |
| CAPTURE_TIME | TIMESTAMP(6) | | Timestamp of capture operation |

# DBA_SSCR_RESTORE

DBA_SSCR_RESTORE displays session state restore statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DB_NAME | VARCHAR2(4000) | | Database name of restored session |
| INST_NAME | VARCHAR2(4000) | | Instance name of restored session |
| INST_ID | NUMBER | | Instance ID of restored session |
| SESSION_ID | NUMBER | | Session ID of restored session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SESSION_SERIAL# | NUMBER | | Session serial number of restored session |
| USER_NAME | VARCHAR2(128) | NOT NULL | User name of restored session |
| SCHEMA_NAME | VARCHAR2(128) | NOT NULL | Schema name of restored session |
| SEQUENCE# | NUMBER | | Sequence number of restore operation |
| RESTORE_MODE | VARCHAR2(7) | | Mode of restore operation |
| RESTORE_SCOPE | VARCHAR2(7) | | Scope of restore operation |
| RESTORE_FORMAT | VARCHAR2(9) | | Format of restore files |
| RESTORE_DIR | VARCHAR2(128) | | Directory object of restore files |
| RESTORE_LOCATOR | RAW(64) | | Locator of master restore file |
| RESTORE_TIME | TIMESTAMP(6) | | Timestamp of restore operation |

# DBA_STAT_EXTENSIONS

DBA_STAT_EXTENSIONS displays information about all optimizer statistics extensions in the database.

**See Also:**

"ALL_STAT_EXTENSIONS"

# DBA_STMT_AUDIT_OPTS

DBA_STMT_AUDIT_OPTS describes current system auditing options across the system and by user.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_NAME | VARCHAR2(128) | | User name if by user auditing; ANY CLIENT if access by a proxy on behalf of a client is being audited; NULL for system-wide auditing |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROXY_NAME | VARCHAR2(128) | | Name of the proxy user which is performing an operation for the client; NULL if the client is performing the operation directly |
| AUDIT_OPTION | VARCHAR2(40) | NOT NULL | Name of the system auditing option |
| SUCCESS | VARCHAR2(10) | | Mode for WHENEVER SUCCESSFUL system auditing |
| FAILURE | VARCHAR2(10) | | Mode for WHENEVER NOT SUCCESSFUL system auditing |

# DBA_STORED_SETTINGS

DBA_STORED_SETTINGS lists information about the persistent parameter settings for stored PL/SQL units for which the current user has execute privileges. It also returns parameter information for all objects in the database and is accessible only to users with the SELECT_CATALOG_ROLE privilege. Its columns are the same as those in "ALL_STORED_SETTINGS".

# 6

# Static Data Dictionary Views: DBA_STREAMS_ADD_COLUMN to USER_ZONEMAPS

This chapter contains the static data dictionary views DBA_STREAMS_ADD_COLUMN through USER_ZONEMAPS.

## DBA_STREAMS_ADD_COLUMN

DBA_STREAMS_ADD_COLUMN displays information about declarative rule-based transformations that add a column to a row logical change record (LCR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | | Name of the rule |
| SCHEMA_NAME | VARCHAR2(128) | | Schema of the column to be added |
| TABLE_NAME | VARCHAR2(128) | | Table of the column to be added |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column to be added |
| COLUMN_VALUE | ANYDATA | | Value of the column to be added |
| COLUMN_TYPE | VARCHAR2(4000) | | Type of the column to be added |
| COLUMN_FUNCTION | VARCHAR2(128) | | Name of the default function used to add a column |
| VALUE_TYPE | VARCHAR2(3) | | Indicates whether to modify the old (OLD), new (NEW), or both (*) values of the LCR |
| PRECEDENCE | NUMBER | | 3 (the execution order relative to other transformations on the same STEP_NUMBER; the smaller number will be executed first) |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

## DBA_STREAMS_ADMINISTRATOR

DBA_STREAMS_ADMINISTRATOR displays information about the users who have been granted privileges to be a Streams administrator by procedures in the DBMS_STREAMS_AUTH package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user who has been granted privileges to be a Streams administrator |
| LOCAL_PRIVILEGES | VARCHAR2(3) | | Indicates whether the user has been granted local Streams administrator privileges (YES) or not (NO) |
| ACCESS_FROM_REMOTE | VARCHAR2(3) | | Indicates whether the user can be used for remote Streams administration through a database link (YES) or not (NO) |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STREAMS_AUTH package

# DBA_STREAMS_COLUMNS

DBA_STREAMS_COLUMNS displays information about all Streams unsupported columns in the database. Its columns are the same as those in ALL_STREAMS_COLUMNS.

**See Also:**

"ALL_STREAMS_COLUMNS"

# DBA_STREAMS_DELETE_COLUMN

DBA_STREAMS_DELETE_COLUMN displays information about declarative rule-based transformations that delete a column from a row logical change record (LCR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | | Name of the rule |
| SCHEMA_NAME | VARCHAR2(128) | | Schema of the column to be deleted |
| TABLE_NAME | VARCHAR2(128) | | Table of the column to be deleted |
| COLUMN_NAME | VARCHAR2(4000) | | Name of the column to delete |
| VALUE_TYPE | VARCHAR2(3) | | Indicates whether to modify the old (OLD), new (NEW), or both (*) values of the LCR |
| PRECEDENCE | NUMBER | | 1 (the execution order relative to other transformations on the same STEP_NUMBER; the smaller number will be executed first) |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

# DBA_STREAMS_GLOBAL_RULES

DBA_STREAMS_GLOBAL_RULES displays information about the global rules created for all Streams capture processes, propagations, and apply processes in the database. Its columns are the same as those in ALL_STREAMS_GLOBAL_RULES.

**See Also:**

"ALL_STREAMS_GLOBAL_RULES"

# DBA_STREAMS_KEEP_COLUMNS

DBA_STREAMS_KEEP_COLUMNS displays information about declarative rule-based transformations that keep a list of columns in a row logical change record (LCR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule which has an associated transformation |
| RULE_NAME | VARCHAR2(128) | | Name of the rule which has an associated transformation |
| SCHEMA_NAME | VARCHAR2(128) | | Schema of the column to be kept |
| TABLE_NAME | VARCHAR2(128) | | Table of the column to be kept |
| COLUMN_NAME | VARCHAR2(4000) | | Column to keep |
| VALUE_TYPE | VARCHAR2(3) | | Indicates whether to keep the old (OLD), new (NEW), or both (*) value of the LCR |
| PRECEDENCE | NUMBER | | 0 (the execution order relative to other transformations on the same STEP_NUMBER; the smaller number will be executed first) |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

# DBA_STREAMS_MESSAGE_CONSUMERS

DBA_STREAMS_MESSAGE_CONSUMERS displays information about all Streams messaging clients in the database. Its columns are the same as those in ALL_STREAMS_MESSAGE_CONSUMERS.

**See Also:**

"ALL_STREAMS_MESSAGE_CONSUMERS"

# DBA_STREAMS_MESSAGE_RULES

DBA_STREAMS_MESSAGE_RULES displays information about all Streams messaging rules in the database. Its columns are the same as those in ALL_STREAMS_MESSAGE_RULES.

**See Also:**

"ALL_STREAMS_MESSAGE_RULES"

# DBA_STREAMS_NEWLY_SUPPORTED

DBA_STREAMS_NEWLY_SUPPORTED displays information about all tables in the database that are newly supported by Streams capture processes. Its columns are the same as those in ALL_STREAMS_NEWLY_SUPPORTED.

**See Also:**

"ALL_STREAMS_NEWLY_SUPPORTED"

# DBA_STREAMS_RENAME_COLUMN

DBA_STREAMS_RENAME_COLUMN displays information about declarative rule-based transformations that rename a column in a row logical change record (LCR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | | Name of the rule |
| SCHEMA_NAME | VARCHAR2(128) | | Schema of the column to be renamed |
| TABLE_NAME | VARCHAR2(128) | | Table of the column to be renamed |
| FROM_COLUMN_NAME | VARCHAR2(4000) | | Column to rename |
| TO_COLUMN_NAME | VARCHAR2(4000) | | New column name |
| VALUE_TYPE | VARCHAR2(3) | | Indicates whether to modify the old (OLD), new (NEW), or both (*) values of the LCR |
| PRECEDENCE | NUMBER | | 2 (the execution order relative to other transformations on the same STEP_NUMBER; the smaller number will be executed first) |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

# DBA_STREAMS_RENAME_SCHEMA

DBA_STREAMS_RENAME_SCHEMA displays information about declarative rule-based transformations that rename a schema in a row logical change record (LCR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | | Name of the rule |
| FROM_SCHEMA_NAME | VARCHAR2(128) | | Schema to be renamed |
| TO_SCHEMA_NAME | VARCHAR2(128) | | New schema name |
| PRECEDENCE | NUMBER | | 5 (the execution order relative to other transformations on the same STEP_NUMBER; the smaller number will be executed first) |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

# DBA_STREAMS_RENAME_TABLE

DBA_STREAMS_RENAME_TABLE displays information about declarative rule-based transformations that rename a table in a row logical change record (LCR).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | | Name of the rule |
| FROM_SCHEMA_NAME | VARCHAR2(128) | | Schema to be renamed |
| TO_SCHEMA_NAME | VARCHAR2(128) | | New schema name |
| FROM_TABLE_NAME | VARCHAR2(128) | | Table to be renamed |
| TO_TABLE_NAME | VARCHAR2(128) | | New table name |
| PRECEDENCE | NUMBER | | 4 (the execution order relative to other transformations on the same STEP_NUMBER; the smaller number will be executed first) |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

# DBA_STREAMS_RULES

DBA_STREAMS_RULES displays information about the rules used by all Streams processes in the database. Its columns are the same as those in ALL_STREAMS_RULES.

**See Also:**

"ALL_STREAMS_RULES"

# DBA_STREAMS_SCHEMA_RULES

DBA_STREAMS_SCHEMA_RULES displays information about the schema rules created for all Streams capture processes, propagations, and apply processes in the database. Its columns are the same as those in ALL_STREAMS_SCHEMA_RULES.

**See Also:**

"ALL_STREAMS_SCHEMA_RULES"

# DBA_STREAMS_SPLIT_MERGE

DBA_STREAMS_SPLIT_MERGE displays information about current Streams automatic split and merge operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ORIGINAL_CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the original capture process |
| CLONED_CAPTURE_NAME | VARCHAR2(128) | | Name of the cloned capture process |
| ORIGINAL_CAPTURE_STATUS | VARCHAR2(8) | | Status of the original capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| CLONED_CAPTURE_STATUS | VARCHAR2(8) | | Status of the cloned capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| ORIGINAL_STREAMS_NAME | VARCHAR2(128) | | Name of the original Oracle Streams component that receives database changes directly from the original capture process. The component is either a propagation or a local apply process. |
| CLONED_STREAMS_NAME | VARCHAR2(128) | | Name of the cloned Oracle Streams component that receives database changes directly from the cloned capture process. The component is either a propagation or a local apply process. |
| STREAMS_TYPE | VARCHAR2(11) | | Type of the component in ORIGINAL_STREAMS_NAME and CLONED_STREAMS_NAME:<br>• PROPAGATION<br>• APPLY |
| RECOVERABLE_SCRIPT_ID | RAW(16) | | Unique ID of the script performing the split or merge operation |
| SCRIPT_STATUS | VARCHAR2(12) | | State of the recoverable script:<br>• GENERATING<br>• NOT EXECUTED<br>• EXECUTING |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • EXECUTED<br>• ERROR |
| ACTION_TYPE | VARCHAR2(7) | | Type of action performed by the script:<br>• SPLIT<br>• MERGE<br>• MONITOR |
| ACTION_THRESHOLD | VARCHAR2(40) | | For SPLIT actions, the threshold set by the split_threshold capture process parameter. For MERGE actions, the threshold set by the merge_threshold capture process parameter. |
| STATUS | VARCHAR2(16) | | Status of the action:<br>• NOTHING TO SPLIT - Not ready to split or does not need to split<br>• ABOUT TO SPLIT<br>• SPLITTING - A split is in progress<br>• SPLIT DONE - A split is done<br>• NOTHING TO MERGE - Not ready to merge<br>• ABOUT TO MERGE<br>• MERGING - A merge is in progress<br>• MERGE DONE - A merge is done<br>• ERROR - An error was returned during a split or merge<br>• NONSPLITTABLE - The original capture is not splittable either because it is disabled, it has more than one publisher to its queue, or it has only one destination for captured messages |
| STATUS_UPDATE_TIME | TIMESTAMP(6) | | Time when status was last updated |
| CREATION_TIME | TIMESTAMP(6) | | Time when the action started |
| LAG | NUMBER | | Time (in seconds) that the cloned capture process lags behind the original capture process |
| JOB_OWNER | VARCHAR2(128) | | Owner of the job that performs the split or merge operation |
| JOB_NAME | VARCHAR2(128) | | Name of the job that performs the split or merge operation |
| JOB_STATE | VARCHAR2(15) | | Current state of the job:<br>• DISABLED<br>• RETRY SCHEDULED<br>• SCHEDULED<br>• RUNNING<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• REMOTE<br>• SUCCEEDED<br>• CHAIN_STALLED |
| JOB_NEXT_RUN_DATE | VARCHAR2(64) | | Next time the job will run |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ERROR_NUMBER | NUMBER | | Error number if the capture process was aborted |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message if the capture process was aborted |

# DBA_STREAMS_SPLIT_MERGE_HIST

DBA_STREAMS_SPLIT_MERGE_HIST displays information about past Streams automatic split and merge operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ORIGINAL_CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the original capture process |
| CLONED_CAPTURE_NAME | VARCHAR2(128) | | Name of the cloned capture process |
| ORIGINAL_QUEUE_OWNER | VARCHAR2(128) | | Owner of the queue used by the original capture process |
| ORIGINAL_QUEUE_NAME | VARCHAR2(128) | | Name of the queue used by the original capture process |
| CLONED_QUEUE_OWNER | VARCHAR2(128) | | Owner of the queue used by the cloned capture process |
| CLONED_QUEUE_NAME | VARCHAR2(128) | | Name of the queue used by the cloned capture process |
| ORIGINAL_CAPTURE_STATUS | VARCHAR2(8) | | Status of the original capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| CLONED_CAPTURE_STATUS | VARCHAR2(8) | | Status of the cloned capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| ORIGINAL_STREAMS_NAME | VARCHAR2(128) | | Name of the original Oracle Streams component that receives database changes directly from the original capture process. The component is either a propagation or a local apply process. |
| CLONED_STREAMS_NAME | VARCHAR2(128) | | Name of the cloned Oracle Streams component that receives database changes directly from the cloned capture process. The component is either a propagation or a local apply process. |
| STREAMS_TYPE | VARCHAR2(11) | | Type of the component in ORIGINAL_STREAMS_NAME and CLONED_STREAMS_NAME:<br>• PROPAGATION<br>• APPLY |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RECOVERABLE_SCRIPT_ID | RAW(16) | | Unique ID of the script performing the split or merge operation |
| SCRIPT_STATUS | VARCHAR2(12) | | State of the recoverable script:<br>• GENERATING<br>• NOT EXECUTED<br>• EXECUTING<br>• EXECUTED<br>• ERROR |
| ACTION_TYPE | VARCHAR2(7) | | Type of action performed by the script:<br>• SPLIT<br>• MERGE<br>• MONITOR |
| ACTION_THRESHOLD | VARCHAR2(40) | | For SPLIT actions, the threshold set by the split_threshold capture process parameter. For MERGE actions, the threshold set by the merge_threshold capture process parameter. |
| STATUS | VARCHAR2(16) | | Status of the action:<br>• NOTHING TO SPLIT - Not ready to split or does not need to split<br>• ABOUT TO SPLIT<br>• SPLITTING - A split is in process<br>• SPLIT DONE - A split is done<br>• NOTHING TO MERGE - Not ready to merge<br>• ABOUT TO MERGE<br>• MERGING - A merge is in process<br>• MERGE DONE - A merge is done<br>• ERROR - An error was returned during a split or merge<br>• NONSPLITTABLE - The original capture is not splittable either because it is disabled, it has more than one publisher to its queue, or it has only one destination for captured messages |
| STATUS_UPDATE_TIME | TIMESTAMP(6) | | Time when status was last updated |
| CREATION_TIME | TIMESTAMP(6) | | Time when the action started |
| LAG | NUMBER | | Time (in seconds) that the cloned capture process lags behind the original capture process |
| JOB_OWNER | VARCHAR2(128) | | Owner of the job that performs the split or merge operation |
| JOB_NAME | VARCHAR2(128) | | Name of the job that performs the split or merge operation |
| ERROR_NUMBER | NUMBER | | Error number if the capture process was aborted |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message if the capture process was aborted |

# DBA_STREAMS_STMT_HANDLERS

DBA_STREAMS_STMT_HANDLERS displays information about all Streams statement DML handlers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HANDLER_NAME | VARCHAR2(128) | NOT NULL | Name of the statement handler |
| HANDLER_COMMENT | VARCHAR2(4000) | | Comment specified with the statement handler |
| CREATION_TIME | TIMESTAMP(6) | | Timestamp for script creation |
| MODIFICATION_TIME | TIMESTAMP(6) | | Timestamp for script modification |

# DBA_STREAMS_STMTS

DBA_STREAMS_STMTS displays information about the statements in all Streams statement DML handlers in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HANDLER_NAME | VARCHAR2(128) | NOT NULL | Name of the statement handler |
| EXECUTION_SEQUENCE | NUMBER | NOT NULL | Execution sequence of the statement |
| STATEMENT | CLOB | | Text of the SQL statement |
| CREATION_TIME | TIMESTAMP(6) | | Timestamp for statement creation |
| MODIFICATION_TIME | TIMESTAMP(6) | | Timestamp for statement modification |

# DBA_STREAMS_TABLE_RULES

DBA_STREAMS_TABLE_RULES displays information about the table rules created for all Streams capture processes, propagations, and apply processes in the database. Its columns are the same as those in ALL_STREAMS_TABLE_RULES.

**See Also:**

"ALL_STREAMS_TABLE_RULES"

# DBA_STREAMS_TP_COMPONENT

DBA_STREAMS_TP_COMPONENT displays information about each Oracle Streams component at each database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPONENT_ID | NUMBER | NOT NULL | ID of the Streams component |
| COMPONENT_NAME | VARCHAR2(4000) | | Name of the Streams component |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPONENT_DB | VARCHAR2(128) | | Database where the Streams component resides |
| COMPONENT_TYPE | VARCHAR2(20) | | Type of the Streams component:<br>• CAPTURE<br>• PROPAGATION SENDER<br>• PROPAGATION RECEIVER<br>• APPLY<br>• QUEUE |
| COMPONENT_CHANGED_TIME | DATE | | Time that the Streams component was last changed |

# DBA_STREAMS_TP_COMPONENT_LINK

DBA_STREAMS_TP_COMPONENT_LINK displays information about how messages flow between Oracle Streams components.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_COMPONENT_ID | NUMBER | NOT NULL | ID of the source Streams component |
| SOURCE_COMPONENT_NAME | VARCHAR2(4000) | | Name of the source Streams component |
| SOURCE_COMPONENT_DB | VARCHAR2(128) | | Database where the source Streams component resides |
| SOURCE_COMPONENT_TYPE | VARCHAR2(20) | | Type of the source Streams component:<br>• CAPTURE<br>• PROPAGATION SENDER<br>• PROPAGATION RECEIVER<br>• APPLY<br>• QUEUE |
| DESTINATION_COMPONENT_ID | NUMBER | NOT NULL | ID of the destination Streams component |
| DESTINATION_COMPONENT_NAME | VARCHAR2(4000) | | Name of the destination Streams component |
| DESTINATION_COMPONENT_DB | VARCHAR2(128) | | Database where the destination Streams component resides |
| DESTINATION_COMPONENT_TYPE | VARCHAR2(20) | | Type of the destination Streams component:<br>• CAPTURE<br>• PROPAGATION SENDER<br>• PROPAGATION RECEIVER<br>• APPLY<br>• QUEUE |
| PATH_ID | NUMBER | NOT NULL | ID of the stream path |
| POSITION | NUMBER | | Position of the link within the stream path |

# DBA_STREAMS_TP_COMPONENT_STAT

DBA_STREAMS_TP_COMPONENT_STAT displays temporary performance statistics and session statistics about each Oracle Streams component.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPONENT_ID | NUMBER | NOT NULL | ID of the Streams component |
| COMPONENT_NAME | VARCHAR2(4000) | | Name of the Streams component |
| COMPONENT_DB | VARCHAR2(128) | | Database where the Streams component resides |
| COMPONENT_TYPE | VARCHAR2(20) | | Type of the Streams component:<br>• CAPTURE<br>• PROPAGATION SENDER<br>• PROPAGATION RECEIVER<br>• APPLY<br>• QUEUE |
| SUB_COMPONENT_TYPE | VARCHAR2(27) | | Type of the Streams subcomponent:<br>• LOGMINER READER<br>• LOGMINER PREPARER<br>• LOGMINER BUILDER<br>• CAPTURE SESSION<br>• PROPAGATION SENDER+RECEIVER<br>• APPLY READER<br>• APPLY COORDINATOR<br>• APPLY SERVER |
| SESSION_ID | NUMBER | | ID of the Streams session for the Streams component |
| SESSION_SERIAL# | NUMBER | | Serial number of the Streams session for the Streams component |
| STATISTIC_TIME | DATE | | Time that the statistic was taken |
| STATISTIC_NAME | VARCHAR2(64) | | Name of the statistic |
| STATISTIC_VALUE | VARCHAR2(4000) | | Value of the statistic |
| STATISTIC_UNIT | VARCHAR2(64) | | Unit of the statistic |
| ADVISOR_RUN_ID | NUMBER | | Logical number (1-based) of the Advisor run |
| ADVISOR_RUN_TIME | DATE | | Time that the Advisor was run |

# DBA_STREAMS_TP_DATABASE

DBA_STREAMS_TP_DATABASE displays information about each database that contains Oracle Streams components.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GLOBAL_NAME | VARCHAR2(128) | NOT NULL | Global name of the Streams database |
| LAST_QUERIED | DATE | NOT NULL | Time that the Streams database was last queried |
| VERSION | VARCHAR2(128) | | Database version of the Streams database |
| COMPATIBILITY | VARCHAR2(128) | | Compatible setting of the Streams database |
| MANAGEMENT_PACK_ACCESS | VARCHAR2(128) | | Management pack access of the Streams database |

# DBA_STREAMS_TP_PATH_BOTTLENECK

DBA_STREAMS_TP_PATH_BOTTLENECK displays temporary information about Oracle Streams components that might be slowing down the flow of messages in a stream path.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PATH_ID | NUMBER | | ID of the stream path |
| COMPONENT_ID | NUMBER | | ID of the bottleneck component |
| COMPONENT_NAME | VARCHAR2(4000) | | Name of the bottleneck component |
| COMPONENT_DB | VARCHAR2(128) | | Database where the bottleneck component resides |
| COMPONENT_TYPE | VARCHAR2(20) | | Type of the bottleneck component:<br>• CAPTURE<br>• PROPAGATION SENDER<br>• PROPAGATION RECEIVER<br>• APPLY<br>• QUEUE |
| TOP_SESSION_ID | NUMBER | | ID of the top session for the bottleneck component |
| TOP_SESSION_SERIAL# | NUMBER | | Serial number of the top session for the bottleneck component |
| ACTION_NAME | VARCHAR2(64) | | Action name for the bottleneck process |
| BOTTLENECK_IDENTIFIED | VARCHAR2(30) | | Indicates whether the bottleneck was identified (YES) or not (NO) |
| ADVISOR_RUN_ID | NUMBER | | Logical number (1-based) of the Advisor run |
| ADVISOR_RUN_TIME | DATE | | Time that the Advisor was run |
| ADVISOR_RUN_REASON | VARCHAR2(4000) | | Reason for the bottleneck analysis result.:<br>• NULL - Bottleneck is identified<br>• PRE-11.1 DATABASE EXISTS - A pre-release 11.1 database exists in the stream path<br>• DIAGNOSTIC PACK REQUIRED - A database in the stream path does not have the diagnostic package installed |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NO BOTTLENECK IDENTIFIED |

# DBA_STREAMS_TP_PATH_STAT

DBA_STREAMS_TP_PATH_STAT displays temporary performance statistics about each stream path that exists in the Oracle Streams topology.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PATH_ID | NUMBER | | ID of the stream path |
| STATISTIC_TIME | DATE | | Time that the statistic was taken |
| STATISTIC_NAME | VARCHAR2(64) | | Name of the statistic |
| STATISTIC_VALUE | NUMBER | | Value of the statistic |
| STATISTIC_UNIT | VARCHAR2(64) | | Unit of the statistic |
| ADVISOR_RUN_ID | NUMBER | | Logical number (1-based) of the Advisor run |
| ADVISOR_RUN_TIME | DATE | | Time that the Advisor was run |

# DBA_STREAMS_TRANSFORM_FUNCTION

DBA_STREAMS_TRANSFORM_FUNCTION displays information about all rule-based transformation functions in the database. Its columns are the same as those in ALL_STREAMS_TRANSFORM_FUNCTION.

**See Also:**

"ALL_STREAMS_TRANSFORM_FUNCTION"

# DBA_STREAMS_TRANSFORMATIONS

DBA_STREAMS_TRANSFORMATIONS displays information about all Streams transformations available on a system, in order of execution.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RULE_OWNER | VARCHAR2(128) | | Owner of the rule |
| RULE_NAME | VARCHAR2(128) | | Name of the rule |
| TRANSFORM_TYPE | VARCHAR2(26) | | Type of the transformation:<br>• DECLARATIVE TRANSFORMATION<br>• SUBSET RULE<br>• CUSTOM TRANSFORMATION |
| FROM_SCHEMA_NAME | VARCHAR2(128) | | Schema to be renamed |
| TO_SCHEMA_NAME | VARCHAR2(128) | | New schema name |
| FROM_TABLE_NAME | VARCHAR2(128) | | Table to be renamed |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TO_TABLE_NAME | VARCHAR2(128) | | New table name |
| SCHEMA_NAME | VARCHAR2(128) | | Schema of the column to be modified |
| TABLE_NAME | VARCHAR2(128) | | Table of the column to be modified |
| FROM_COLUMN_NAME | VARCHAR2(4000) | | Column to be renamed |
| TO_COLUMN_NAME | VARCHAR2(4000) | | New column name |
| COLUMN_NAME | VARCHAR2(4000) | | Column to add or delete |
| COLUMN_VALUE | ANYDATA | | Value of the column to add |
| COLUMN_TYPE | VARCHAR2(4000) | | Type of the new column |
| COLUMN_FUNCTION | VARCHAR2(128) | | Name of the default function used to add a column |
| VALUE_TYPE | VARCHAR2(3) | | Indicates whether to modify the old (OLD), new (NEW), or both (*) values of the LCR |
| USER_FUNCTION_NAME | VARCHAR2(4000) | | Name of the user-defined transformation function to run |
| SUBSETTING_OPERATION | VARCHAR2(6) | | DML operation for row subsetting:<br>• INSERT<br>• UPDATE<br>• DELETE |
| DML_CONDITION | VARCHAR2(4000) | | Row subsetting condition |
| DECLARATIVE_TYPE | VARCHAR2(13) | | Type of declarative transform:<br>• KEEP COLUMNS<br>• DELETE COLUMN<br>• RENAME COLUMN<br>• ADD COLUMN<br>• RENAME TABLE<br>• RENAME SCHEMA |
| PRECEDENCE | NUMBER | | Execution order relative to other declarative transformations on the same STEP_NUMBER |
| STEP_NUMBER | NUMBER | | Order in which this transformation should be executed |

# DBA_STREAMS_UNSUPPORTED

DBA_STREAMS_UNSUPPORTED displays information about all tables in the database that are not supported by Streams in this release of the Oracle Database. Its columns are the same as those in ALL_STREAMS_UNSUPPORTED.

**See Also:**

"ALL_STREAMS_UNSUPPORTED"

## DBA_SUBPART_COL_STATISTICS

DBA_SUBPART_COL_STATISTICS provides column statistics and histogram information for all subpartitions in the database. Its columns are the same as those in "ALL_SUBPART_COL_STATISTICS".

## DBA_SUBPART_HISTOGRAMS

DBA_SUBPART_HISTOGRAMS lists actual histogram data (end-points per histogram) for histograms on all table subpartitions in the database. Its columns are the same as those in "ALL_SUBPART_HISTOGRAMS".

## DBA_SUBPART_KEY_COLUMNS

DBA_SUBPART_KEY_COLUMNS lists subpartitioning key columns for all composite-partitioned tables (and local indexes on composite-partitioned tables) in the database. Its columns are the same as those in ALL_SUBPART_KEY_COLUMNS.

**See Also:**

"ALL_SUBPART_KEY_COLUMNS"

## DBA_SUBPARTITION_TEMPLATES

DBA_SUBPARTITION_TEMPLATES describes all subpartition templates in the database. Its columns are the same as those in ALL_SUBPARTITION_TEMPLATES.

**See Also:**

"ALL_SUBPARTITION_TEMPLATES"

## DBA_SUBSCR_REGISTRATIONS

DBA_SUBSCR_REGISTRATIONS displays information about all subscription registrations in the database.

### Related View

USER_SUBSCR_REGISTRATIONS displays information about the subscription registrations owned by the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REG_ID | NUMBER | | Registration ID |
| SUBSCRIPTION_NAME | VARCHAR2(128) | NOT NULL | Name of the subscription registration. The subscription name is of the form *schema.queue* if the registration is for a single consumer queue |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | or *schema.queue:consumer_name* if the registration is for a multiconsumer queue. |
| LOCATION_NAME | VARCHAR2(256) | NOT NULL | Location endpoint of the registration |
| USER# | NUMBER | NOT NULL | Internally generated user ID |
| USER_CONTEXT | RAW(128) | | Context the user provided during registration of PL/SQL registrations or an internally generated context for OCI registrations |
| CONTEXT_SIZE | NUMBER | | Size of the context |
| NAMESPACE | VARCHAR2(9) | | Namespace of the subscription registration:<br>• ANONYMOUS<br>• AQ<br>• DBCHANGE |
| PRESENTATION | VARCHAR2(7) | | Presentation format of notifications:<br>• DEFAULT - Binary<br>• XML |
| VERSION | VARCHAR2(8) | | Database version:<br>• 8.1.6<br>• 10.2<br>• 11.1 |
| STATUS | VARCHAR2(8) | | Status of the registration:<br>• DB REG - Database registration<br>• LDAP REG - LDAP registration |
| ANY_CONTEXT | ANYDATA | | AnyData user context (used for Streams replication registrations) |
| CONTEXT_TYPE | NUMBER | | Type of the user context |
| QOSFLAGS | VARCHAR2(64) | | Quality of service of the registration:<br>• RELIABLE - Reliable notifications persist across instance and database restarts<br>• PAYLOAD - Payload delivery is required. It is only supported for client notification and only for RAW queues.<br>• PURGE_ON_NTFN - Registration is to be purged automatically when the first notification is delivered to this registration location |
| PAYLOAD_CALLBACK | VARCHAR2(4000) | | Any callback registered to serialize the notification payload |
| TIMEOUT | TIMESTAMP(6) | | Registration timeout |
| REG_TIME | TIMESTAMP(6) WITH TIME ZONE | | Time of the registration |
| NTFN_GROUPING_CLASS | VARCHAR2(4) | | Notification grouping class |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NTFN_GROUPING_VALUE | NUMBER | | Notification grouping value |
| NTFN_GROUPING_TYPE | VARCHAR2(7) | | Notification grouping type:<br>• SUMMARY<br>• LAST |
| NTFN_GROUPING_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Notification grouping start time |
| NTFN_GROUPING_REPEAT_COUNT | VARCHAR2(40) | | Notification grouping repeat count, or FOREVER |

**See Also:**

"USER_SUBSCR_REGISTRATIONS"

# DBA_SUPPLEMENTAL_LOGGING

DBA_SUPPLEMENTAL_LOGGING provides information about supplemental logging for a pluggable database (PDB) in a multitenant container database (CDB).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MINIMAL | VARCHAR2(3) | | Identifies whether minimal supplemental logging is on (YES or NO) |
| PRIMARY_KEY | VARCHAR2(3) | | Identifies whether primary key supplemental logging is on (YES or NO) |
| UNIQUE_INDEX | VARCHAR2(3) | | Identifies whether unique column supplemental logging is on (YES or NO) |
| FOREIGN_KEY | VARCHAR2(3) | | Identifies whether foreign key supplemental logging is on (YES or NO) |
| ALL_COLUMN | VARCHAR2(3) | | Identifies whether all column supplemental logging is on (YES or NO) |
| PROCEDURAL | VARCHAR2(3) | | Identifies whether supplemental logging for procedural replication is on (YES or NO) |

**See Also:**

- *Oracle Database Utilities* for more information about supplemental logging
- "V$DATABASE" for information about supplemental logging in a CDB

# DBA_SYNC_CAPTURE

DBA_SYNC_CAPTURE displays information about all synchronous capture processes in the database. Its columns are the same as those in ALL_SYNC_CAPTURE.

**See Also:**

"ALL_SYNC_CAPTURE"

## DBA_SYNC_CAPTURE_PREPARED_TABS

DBA_SYNC_CAPTURE_PREPARED_TABS displays information about all tables in the database that are prepared for synchronous capture instantiation. Its columns are the same as those in ALL_SYNC_CAPTURE_PREPARED_TABS.

**See Also:**

"ALL_SYNC_CAPTURE_PREPARED_TABS"

## DBA_SYNC_CAPTURE_TABLES

DBA_SYNC_CAPTURE_TABLES displays information about all tables in the database that are captured by synchronous Streams captures. Its columns are the same as those in ALL_SYNC_CAPTURE_TABLES.

**See Also:**

"ALL_SYNC_CAPTURE_TABLES"

## DBA_SYNONYMS

DBA_SYNONYMS describes all synonyms in the database. Its columns are the same as those in ALL_SYNONYMS.

**See Also:**

"ALL_SYNONYMS"

## DBA_SYS_PRIVS

DBA_SYS_PRIVS describes system privileges granted to users and roles.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Grantee name, user, or role receiving the grant |
| PRIVILEGE | VARCHAR2(40) | | System privilege |
| ADMIN_OPTION | VARCHAR2(3) | | Indicates whether the grant was with the ADMIN option (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • NO if the privilege was granted locally (CONTAINER=ALL was not used) |

**See Also:**

"USER_SYS_PRIVS"

# DBA_TAB_COL_STATISTICS

DBA_TAB_COL_STATISTICS contains column statistics and histogram information extracted from "DBA_TAB_COLUMNS". Its columns are the same as those in "ALL_TAB_COL_STATISTICS".

# DBA_TAB_COLS

DBA_TAB_COLS describes the columns of all tables, views, and clusters in the database. This view differs from "DBA_TAB_COLUMNS" in that system-generated hidden columns and invisible columns, which are user-generated hidden columns, are not filtered out.

**Note:**

See *Oracle Database Administrator's Guide* for information about invisible columns.

Its columns are the same as those in "ALL_TAB_COLS", except for the SENSITIVE_COLUMN. To gather statistics for this view, use the DBMS_STATS package.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table, view, or cluster |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table, view, or cluster |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name |
| DATA_TYPE | VARCHAR2(128) | | data type of the column |
| DATA_TYPE_MOD | VARCHAR2(3) | | data type modifier of the column |
| DATA_TYPE_OWNER | VARCHAR2(128) | | Owner of the data type of the column |
| DATA_LENGTH | NUMBER | NOT NULL | Length of the column (in bytes) |
| DATA_PRECISION | NUMBER | | Decimal precision for NUMBER data type; binary precision for FLOAT data type; NULL for all other data types |
| DATA_SCALE | NUMBER | | Digits to the right of the decimal point in a number |
| NULLABLE | VARCHAR2(1) | | Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | the column or if the column is part of a PRIMARY KEY. |
| COLUMN_ID | NUMBER | | Sequence number of the column as created |
| DEFAULT_LENGTH | NUMBER | | Length of the default value for the column |
| DATA_DEFAULT | LONG | | Default value for the column |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column[1] |
| HIGH_VALUE | RAW(1000) | | High value in the column[1] |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/NUM_DISTINCT.[1] |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| NUM_BUCKETS | NUMBER | | Number of buckets in the histogram for the column<br>**Note:** The number of buckets in a histogram is specified in the SIZE parameter of the ANALYZE SQL statement. However, Oracle Database does not create a histogram with more buckets than the number of rows in the sample. Also, if the sample contains any values that are very repetitious, Oracle Database creates the specified number of buckets, but the value indicated by this column may be smaller because of an internal compression algorithm. |
| LAST_ANALYZED | DATE | | Date on which this column was most recently analyzed |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this column |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Name of the character set:<br>• CHAR_CS<br>• NCHAR_CS |
| CHAR_COL_DECL_LENGTH | NUMBER | | Declaration length of the character type column |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| CHAR_LENGTH | NUMBER | | Displays the length of the column in characters. This value only applies to the following data types:<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| CHAR_USED | VARCHAR2(1) | | Indicates that the column uses BYTE length semantics (B) or CHAR length semantics (C), or whether the data type is not any of the following (NULL):<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| V80_FMT_IMAGE | VARCHAR2(3) | | Indicates whether the column data is in release 8.0 image format (YES) or not (NO) |
| DATA_UPGRADED | VARCHAR2(3) | | Indicates whether the column data has been upgraded to the latest type version format (YES) or not (NO) |
| HIDDEN_COLUMN | VARCHAR2(3) | | Indicates whether the column is a hidden column (YES) or not (NO) |
| VIRTUAL_COLUMN | VARCHAR2(3) | | Indicates whether the column is a virtual column (YES) or not (NO) |
| SEGMENT_COLUMN_ID | NUMBER | | Sequence number of the column in the segment |
| INTERNAL_COLUMN_ID | NUMBER | NOT NULL | Internal sequence number of the column |
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE<br>• FREQUENCY<br>• HEIGHT BALANCED |
| QUALIFIED_COL_NAME | VARCHAR2(4000) | | Qualified column name |
| USER_GENERATED | VARCHAR2(3) | | Indicates whether the column is a user-generated column (YES) or a system-generated column (NO) |
| DEFAULT_ON_NULL | VARCHAR2(3) | | Indicates whether the column has DEFAULT ON NULL semantics (YES) or not (NO) |
| IDENTITY_COLUMN | VARCHAR2(3) | | Indicates whether this is an identity column (YES) or not (NO) |
| SENSITIVE_COLUMN | VARCHAR2(3) | | Indicates whether this is a sensitive column (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVALUATION_EDITION | VARCHAR2(128) | | Name of the edition in which editioned objects referenced in an expression column are resolved |
| UNUSABLE_BEFORE | VARCHAR2(128) | | Name of the oldest edition in which the index may be used as part of a query plan |
| UNUSABLE_BEGINNING | VARCHAR2(128) | | Name of the edition for which the index may not be used as part of a query plan in this edition or any of its descendants |

[1] These columns remain for backward compatibility with Oracle7. This information is now in the [TAB|PART}_COL_STATISTICS views.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# DBA_TAB_COLUMNS

DBA_TAB_COLUMNS describes the columns of all tables, views, and clusters in the database. Its columns are the same as those in "ALL_TAB_COLUMNS", except for SENSITIVE_COLUMN. To gather statistics for this view, use the DBMS_STATS package.

This view filters out system-generated hidden columns and invisible columns, which are user-generated hidden columns. The DBA_TAB_COLS view does not filter out hidden columns and invisible columns.

**See Also:**

For more information about invisible columns:

- "DBA_TAB_COLS"
- *Oracle Database Administrator's Guide*

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the table, view, or cluster |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the table, view, or cluster |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Column name |
| DATA_TYPE | VARCHAR2(128) | | data type of the column |
| DATA_TYPE_MOD | VARCHAR2(3) | | data type modifier of the column |
| DATA_TYPE_OWNER | VARCHAR2(128) | | Owner of the data type of the column |
| DATA_LENGTH | NUMBER | NOT NULL | Length of the column (in bytes) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DATA_PRECISION | NUMBER | | Decimal precision for NUMBER data type; binary precision for FLOAT data type; NULL for all other data types |
| DATA_SCALE | NUMBER | | Digits to the right of the decimal point in a number |
| NULLABLE | VARCHAR2(1) | | Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. The constraint should be in an ENABLE VALIDATE state. |
| COLUMN_ID | NUMBER | | Sequence number of the column as created |
| DEFAULT_LENGTH | NUMBER | | Length of the default value for the column |
| DATA_DEFAULT | LONG | | Default value for the column |
| NUM_DISTINCT | NUMBER | | Number of distinct values in the column |
| LOW_VALUE | RAW(1000) | | Low value in the column[1] |
| HIGH_VALUE | RAW(1000) | | High value in the column[1] |
| DENSITY | NUMBER | | If a histogram is available on COLUMN_NAME, then this column displays the selectivity of a value that spans fewer than 2 endpoints in the histogram. It does not represent the selectivity of values that span 2 or more endpoints.<br>If a histogram is not available on COLUMN_NAME, then the value of this column is 1/NUM_DISTINCT.[1] |
| NUM_NULLS | NUMBER | | Number of NULLs in the column |
| NUM_BUCKETS | NUMBER | | Number of buckets in the histogram for the column<br>**Note:** The number of buckets in a histogram is specified in the SIZE parameter of the ANALYZE SQL statement. However, Oracle Database does not create a histogram with more buckets than the number of rows in the sample. Also, if the sample contains any values that are very repetitious, Oracle Database creates the specified number of buckets, but the value indicated by this column may be smaller because of an internal compression algorithm. |
| LAST_ANALYZED | DATE | | Date on which this column was most recently analyzed |
| SAMPLE_SIZE | NUMBER | | Sample size used in analyzing this column |
| CHARACTER_SET_NAME | VARCHAR2(44) | | Name of the character set:<br>• CHAR_CS<br>• NCHAR_CS |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CHAR_COL_DECL_LENGTH | NUMBER | | Declaration length of the character type column |
| GLOBAL_STATS | VARCHAR2(3) | | Indicates whether statistics were gathered for the table as a whole (global statistics) (YES) or they were not gathered (NO). Note that for partitioned tables, global statistics and partition statistics are estimated from underlying partition and subpartition statistics, respectively, when set to NO. |
| USER_STATS | VARCHAR2(3) | | Indicates whether statistics were entered directly by the user (YES) or not (NO) |
| AVG_COL_LEN | NUMBER | | Average length of the column (in bytes) |
| CHAR_LENGTH | NUMBER | | Displays the length of the column in characters. This value only applies to the following data types:<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| CHAR_USED | VARCHAR2(1) | | Indicates that the column uses BYTE length semantics (B) or CHAR length semantics (C), or whether the data type is not any of the following (NULL):<br>• CHAR<br>• VARCHAR2<br>• NCHAR<br>• NVARCHAR2 |
| V80_FMT_IMAGE | VARCHAR2(3) | | Indicates whether the column data is in release 8.0 image format (YES) or not (NO) |
| DATA_UPGRADED | VARCHAR2(3) | | Indicates whether the column data has been upgraded to the latest type version format (YES) or not (NO) |
| HISTOGRAM | VARCHAR2(15) | | Indicates existence/type of histogram:<br>• NONE<br>• FREQUENCY<br>• HEIGHT BALANCED |
| DEFAULT_ON_NULL | VARCHAR2(3) | | Indicates whether the column has DEFAULT ON NULL semantics (YES) or not (NO) |
| IDENTITY_COLUMN | VARCHAR2(3) | | Indicates whether this is an identity column (YES) or not (NO) |
| SENSITIVE_COLUMN | VARCHAR2(3) | | Indicates whether this is a sensitive column (YES) or not (NO) |
| EVALUATION_EDITION | VARCHAR2(128) | | Name of the edition in which editioned objects referenced in an expression column are resolved |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| UNUSABLE_BEFORE | VARCHAR2(128) | | Name of the oldest edition in which the index may be used as part of a query plan |
| UNUSABLE_BEGINNING | VARCHAR2(128) | | Name of the edition for which the index may not be used as part of a query plan in this edition or any of its descendants |

[1] These columns remain for backward compatibility with Oracle7. This information is now in the [TAB|PART}_COL_STATISTICS views.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# DBA_TAB_COMMENTS

DBA_TAB_COMMENTS displays comments on all tables and views in the database. Its columns are the same as those in ALL_TAB_COMMENTS.

**See Also:**

"ALL_TAB_COMMENTS"

# DBA_TAB_HISTGRM_PENDING_STATS

DBA_TAB_HISTGRM_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions in the database. Its columns are the same as those in "ALL_TAB_HISTGRM_PENDING_STATS".

# DBA_TAB_HISTOGRAMS

DBA_TAB_HISTOGRAMS describes histograms on columns of all tables in the database. Its columns are the same as those in "ALL_TAB_HISTOGRAMS".

# DBA_TAB_IDENTITY_COLS

DBA_TAB_IDENTITY_COLS describes all table identity columns. Its columns are the same as those in ALL_TAB_IDENTITY_COLS.

**See Also:**

"ALL_TAB_IDENTITY_COLS"

# DBA_TAB_MODIFICATIONS

DBA_TAB_MODIFICATIONS describes modifications to all tables in the database that have been modified since the last time statistics were gathered on the tables. Its columns are the same as those in "ALL_TAB_MODIFICATIONS".

**Note:**

This view is populated only for tables with the MONITORING attribute. It is intended for statistics collection over a long period of time. For performance reasons, the Oracle Database does not populate this view immediately when the actual modifications occur. Run the FLUSH_DATABASE_MONITORING_INFO procedure in the DBMS_STATS PL/SQL package to populate this view with the latest information. The ANALYZE_ANY system privilege is required to run this procedure.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the FLUSH_DATABASE_MONITORING_INFO procedure in the DBMS_STATS PL/SQL package

# DBA_TAB_PARTITIONS

DBA_TAB_PARTITIONS displays partition-level partitioning information, partition storage parameters, and partition statistics generated by the DBMS_STATS package for all partitions in the database. Its columns are the same as those in "ALL_TAB_PARTITIONS".

# DBA_TAB_PENDING_STATS

DBA_TAB_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions in the database. Its columns are the same as those in "ALL_TAB_PENDING_STATS".

# DBA_TAB_PRIVS

DBA_TAB_PRIVS describes all object grants in the database.

### Related View

USER_TAB_PRIVS describes the object grants for which the current user is the object owner, grantor, or grantee.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | | Name of the user or role to whom access was granted |
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object. The object can be any object, including tables, packages, indexes, sequences, and so on. |
| GRANTOR | VARCHAR2(128) | | Name of the user who performed the grant |
| PRIVILEGE | VARCHAR2(40) | | Privilege on the object |
| GRANTABLE | VARCHAR2(3) | | Indicates whether the privilege was granted with the GRANT OPTION (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HIERARCHY | VARCHAR2(3) | | Indicates whether the privilege was granted with the HIERARCHY OPTION (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |
| TYPE | VARCHAR2(24) | | Type of the object |

**See Also:**

"USER_TAB_PRIVS"

# DBA_TAB_STATISTICS

DBA_TAB_STATISTICS displays optimizer statistics for all tables in the database. Its columns are the same as those in ALL_TAB_STATISTICS.

**See Also:**

"ALL_TAB_STATISTICS"

# DBA_TAB_STAT_PREFS

DBA_TAB_STAT_PREFS displays information about statistics preferences for all tables in the database. Its columns are the same as those in "ALL_TAB_STAT_PREFS".

# DBA_TAB_STATS_HISTORY

DBA_TAB_STATS_HISTORY provides a history of table statistics modifications for all tables in the database. Its columns are the same as those in "ALL_TAB_STATS_HISTORY".

# DBA_TAB_SUBPARTITIONS

DBA_TAB_SUBPARTITIONS displays, for each table subpartition, the subpartition name, name of the table and partition to which it belongs, its storage attributes, and statistics generated by the DBMS_STATS package. Its columns are the same as those in "ALL_TAB_SUBPARTITIONS".

# DBA_TABLES

DBA_TABLES describes all relational tables in the database. Its columns are the same as those in ALL_TABLES. To gather statistics for this view, use the DBMS_STATS package.

**See Also:**

"ALL_TABLES"

# DBA_TABLESPACE_GROUPS

DBA_TABLESPACE_GROUPS describes all tablespace groups in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GROUP_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace group |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace |

# DBA_TABLESPACE_THRESHOLDS

DBA_TABLESPACE_THRESHOLDS describes space utilization threshold settings for all tablespaces in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | | Tablespace name |
| CONTENTS | VARCHAR2(9) | | Tablespace contents:<br>• UNDO<br>• PERMANENT<br>• TEMPORARY |
| EXTENT_MANAGEMENT | VARCHAR2(10) | | Indicates whether the extents in the tablespace are dictionary managed (DICTIONARY) or locally managed (LOCAL) |
| THRESHOLD_TYPE | VARCHAR2(8) | | Indicates whether the threshold value is derived from a DEFAULT threshold or an EXPLICIT threshold |
| METRICS_NAME | VARCHAR2(64) | | Name of the metric for which the threshold is set |
| WARNING_OPERATOR | VARCHAR2(12) | | Relational operator for warning thresholds:<br>• GT<br>• EQ<br>• LT<br>• LE<br>• GE<br>• CONTAINS<br>• NE<br>• DO_NOT_CHECK |
| WARNING_VALUE | VARCHAR2(256) | | Warning threshold value |
| CRITICAL_OPERATOR | VARCHAR2(12) | | Relational operator for critical thresholds:<br>• GT<br>• EQ<br>• LT<br>• LE<br>• GE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • CONTAINS<br>• NE<br>• DO_NOT_CHECK |
| CRITICAL_VALUE | VARCHAR2(256) | | Critical threshold value |

# DBA_TABLESPACE_USAGE_METRICS

DBA_TABLESPACE_USAGE_METRICS describes tablespace usage metrics for all types of tablespaces, including permanent, temporary, and undo tablespaces.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | | Tablespace name |
| USED_SPACE | NUMBER | | Total space consumed by the tablespace, in database blocks |
| TABLESPACE_SIZE | NUMBER | | Maximum size of the tablespace, in database blocks |
| USED_PERCENT | NUMBER | | Percentage of used space, as a function of the maximum possible tablespace size |

# DBA_TABLESPACES

DBA_TABLESPACES describes all tablespaces in the database.

### Related View

USER_TABLESPACES describes the tablespaces accessible to the current user. This view does not display the PLUGGED_IN column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace |
| BLOCK_SIZE | NUMBER | NOT NULL | Tablespace block size (in bytes) |
| INITIAL_EXTENT | NUMBER | | Default initial extent size (in bytes) |
| NEXT_EXTENT | NUMBER | | Default incremental extent size (in bytes) |
| MIN_EXTENTS | NUMBER | NOT NULL | Default minimum number of extents |
| MAX_EXTENTS | NUMBER | | Default maximum number of extents |
| MAX_SIZE | NUMBER | | Default maximum size of segments (in Oracle blocks) |
| PCT_INCREASE | NUMBER | | Default percent increase for extent size |
| MIN_EXTLEN | NUMBER | | Minimum extent size for this tablespace (in bytes) |
| STATUS | VARCHAR2(9) | | Tablespace status:<br>• ONLINE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • OFFLINE<br>• READ ONLY |
| CONTENTS | VARCHAR2(9) | | Tablespace contents:<br>• UNDO<br>• PERMANENT<br>• TEMPORARY |
| LOGGING | VARCHAR2(9) | | Default logging attribute:<br>• LOGGING<br>• NOLOGGING |
| FORCE_LOGGING | VARCHAR2(3) | | Indicates whether the tablespace is under force logging mode (YES) or not (NO) |
| EXTENT_MANAGEMENT | VARCHAR2(10) | | Indicates whether the extents in the tablespace are dictionary managed (DICTIONARY) or locally managed (LOCAL) |
| ALLOCATION_TYPE | VARCHAR2(9) | | Type of extent allocation in effect for the tablespace:<br>• SYSTEM<br>• UNIFORM<br>• USER |
| PLUGGED_IN | VARCHAR2(3) | | Indicates whether the tablespace is plugged in (YES) or not (NO) |
| SEGMENT_SPACE_MANAGEMENT | VARCHAR2(6) | | Indicates whether the free and used segment space in the tablespace is managed using free lists (MANUAL) or bitmaps (AUTO) |
| DEF_TAB_COMPRESSION | VARCHAR2(8) | | Indicates whether default table compression is enabled (ENABLED) or not (DISABLED)<br>**Note:** Enabling default table compression indicates that all tables in the tablespace will be created with table compression enabled unless otherwise specified. |
| RETENTION | VARCHAR2(11) | | Undo tablespace retention:<br>• GUARANTEE - Tablespace is an undo tablespace with RETENTION specified as GUARANTEE<br>A RETENTION value of GUARANTEE indicates that unexpired undo in all undo segments in the undo tablespace should be retained even if it means that forward going operations that need to generate undo in those segments fail.<br>• NOGUARANTEE - Tablespace is an undo tablespace with RETENTION specified as NOGUARANTEE<br>• NOT APPLY - Tablespace is not an undo tablespace |
| BIGFILE | VARCHAR2(3) | | Indicates whether the tablespace is a bigfile tablespace (YES) or a smallfile tablespace (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PREDICATE_EVALUATION | VARCHAR2(7) | | Indicates whether predicates are evaluated by host (HOST) or by storage (STORAGE) |
| ENCRYPTED | VARCHAR2(3) | | Indicates whether the tablespace is encrypted (YES) or not (NO) |
| COMPRESS_FOR | VARCHAR2(30) | | Default compression for what kind of operations:<br>• BASIC<br>• ADVANCED<br>• QUERY LOW<br>• QUERY HIGH[1]<br>• ARCHIVE LOW[1]<br>• ARCHIVE HIGH[1]<br>• NULL |
| DEF_INMEMORY | VARCHAR2(8) | | Indicates whether the In-Memory Column Store (IM column store) is by default enabled (ENABLED) or disabled (DISABLED) for tables in this tablespace |
| DEF_INMEMORY_PRIORITY[2] | VARCHAR2(8) | | Indicates the default priority for In-Memory Column Store (IM column store) population for this tablespace. Possible values:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE<br>• NULL |
| DEF_INMEMORY_DISTRIBUTE[2] | VARCHAR2(15) | | Indicates how the IM column store is distributed by default for this tablespace in an Oracle Real Application Clusters (Oracle RACE) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| DEF_INMEMORY_COMPRESSION[2] | VARCHAR2(17) | | Indicates the default compression level for the IM column store for this tablespace:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ]<br>• NULL |
| DEF_INMEMORY_DUPLICATE[2] | VARCHAR2(13) | | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |

[1] Hybrid Columnar Compression is a feature of the Enterprise Edition of Oracle Database that is dependent on the underlying storage system. See *Oracle Database Concepts* for more information.

[2] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"USER_TABLESPACES"

# DBA_TEMP_FILES

DBA_TEMP_FILES describes all temporary files (tempfiles) in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_NAME | VARCHAR2(513) | | Name of the database temp file |
| FILE_ID | NUMBER | | File identifier number of the database temp file |
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace to which the file belongs |
| BYTES | NUMBER | | Size of the file (in bytes) |
| BLOCKS | NUMBER | | Size of the file (in Oracle blocks) |
| STATUS | VARCHAR2(7) | | File status:<br>• OFFLINE<br>• ONLINE<br>• UNKNOWN |
| RELATIVE_FNO | NUMBER | | Tablespace-relative file number |
| AUTOEXTENSIBLE | VARCHAR2(3) | | Indicates whether the file is autoextensible (YES) or not (NO) |
| MAXBYTES | NUMBER | | maximum size of the file (in bytes) |
| MAXBLOCKS | NUMBER | | Maximum size of the file (in Oracle blocks) |
| INCREMENT_BY | NUMBER | | Default increment for autoextension (in Oracle blocks) |
| USER_BYTES | NUMBER | | Size of the useful portion of the file (in bytes) |
| USER_BLOCKS | NUMBER | | Size of the useful portion of the file (in Oracle blocks) |

# DBA_TEMP_FREE_SPACE

DBA_TEMP_FREE_SPACE displays temporary space usage information at tablespace level.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace |
| TABLESPACE_SIZE | NUMBER | | Total size of the tablespace, in bytes |
| ALLOCATED_SPACE | NUMBER | | Total allocated space, in bytes, including space that is currently allocated and used and space that is currently allocated and available for reuse |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FREE_SPACE | NUMBER | | Total free space available, in bytes, including space that is currently allocated and available for reuse and space that is currently unallocated |

# DBA_THRESHOLDS

DBA_THRESHOLDS describes all thresholds.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| METRICS_NAME | VARCHAR2(64) | | Metrics name |
| WARNING_OPERATOR | VARCHAR2(12) | | Relational operator for warning thresholds:<br>• GT<br>• EQ<br>• LT<br>• LE<br>• GE<br>• CONTAINS<br>• NE<br>• DO NOT CHECK<br>• DO_NOT_CHECK |
| WARNING_VALUE | VARCHAR2(256) | | Warning threshold value |
| CRITICAL_OPERATOR | VARCHAR2(12) | | Relational operator for critical thresholds:<br>• GT<br>• EQ<br>• LT<br>• LE<br>• GE<br>• CONTAINS<br>• NE<br>• DO NOT CHECK<br>• DO_NOT_CHECK |
| CRITICAL_VALUE | VARCHAR2(256) | | Critical threshold value |
| OBSERVATION_PERIOD | NUMBER | | Observation period length (in minutes) |
| CONSECUTIVE_OCCURRENCES | NUMBER | | Number of occurrences before an alert is issued |
| INSTANCE_NAME | VARCHAR2(16) | | Instance name; NULL for database-wide alerts |
| OBJECT_TYPE | VARCHAR2(64) | | Object type:<br>• SYSTEM<br>• SERVICE<br>• EVENT_CLASS<br>• TABLESPACE<br>• FILE |
| OBJECT_NAME | VARCHAR2(513) | | Name of the object for which the threshold is set |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATUS | VARCHAR2(7) | | Indicates whether the threshold is applicable on a valid object (VALID) or not (INVALID)<br>Thresholds for non-tablespace metrics can only be set in ROOT and apply to a CDB as a whole. Any pre-existing non-tablespace thresholds that may exist in a PDB have a status of INVALID in the DBA_THRESHOLDS view. You can remove these threshold settings using the DBMS_SERVER_ALERT.SET_THRESHOLD API.<br>See *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SERVER_ALERT.SET_THRESHOLD API. |

## DBA_TRANSFORMATIONS

DBA_TRANSFORMATIONS displays information about all transformations in the database. These transformations can be specified with Advanced Queuing operations such as enqueue, dequeue, and subscribe to automatically integrate transformations in AQ messaging. Its columns are the same as those in ALL_TRANSFORMATIONS.

**See Also:**

"ALL_TRANSFORMATIONS"

## DBA_TRIGGER_COLS

DBA_TRIGGER_COLS describes the use of columns in all triggers in the database. Its columns are the same as those in ALL_TRIGGER_COLS.

**See Also:**

"ALL_TRIGGER_COLS"

## DBA_TRIGGER_ORDERING

DBA_TRIGGER_ORDERING describes all triggers in the database that have FOLLOWS or PRECEDES ordering. Its columns are the same as those in ALL_TRIGGER_ORDERING.

**See Also:**

"ALL_TRIGGER_ORDERING"

## DBA_TRIGGERS

DBA_TRIGGERS describes all triggers in the database. Its columns are the same as those in ALL_TRIGGERS.

**See Also:**

"ALL_TRIGGERS"

# DBA_TS_QUOTAS

DBA_TS_QUOTAS describes tablespace quotas for all users.

### Related View

USER_TS_QUOTAS describes tablespace quotas for the current user. This view does not display the USERNAME column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Tablespace name |
| USERNAME | VARCHAR2(128) | NOT NULL | User with resource rights on the tablespace |
| BYTES | NUMBER | | Number of bytes charged to the user |
| MAX_BYTES | NUMBER | | User's quota in bytes, or -1 if no limit |
| BLOCKS | NUMBER | | Number of Oracle blocks charged to the user |
| MAX_BLOCKS | NUMBER | | User's quota in Oracle blocks, or -1 if no limit |
| DROPPED | VARCHAR2(3) | | Whether the tablespace has been dropped |

**See Also:**

"USER_TS_QUOTAS"

# DBA_TSDP_IMPORT_ERRORS

DBA_TSDP_IMPORT_ERRORS shows information about the errors encountered during import of the Transparent Sensitive Data Protection discovery result. This error information corresponds to the last import of the discovery result done using the DBMS_TSDP_MANAGE.IMPORT_DISCOVERY_RESULT API.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ERROR_CODE | NUMBER | NOT NULL | The ORA error code of the error encountered |
| SCHEMA_NAME | VARCHAR2(128) | | The schema corresponding to the error |
| TABLE_NAME | VARCHAR2(128) | | The table corresponding to the error |
| COLUMN_NAME | VARCHAR2(128) | | The column corresponding to the error |
| SENSITIVE_TYPE | VARCHAR2(128) | | The sensitive type corresponding to the error |

**See Also:**

- *Oracle Database Security Guide* for more information about using Transparent Sensitive Data Protection
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_TSDP_MANAGE.IMPORT_DISCOVERY_RESULT` procedure

# DBA_TSDP_POLICY_CONDITION

`DBA_TSDP_POLICY_CONDITION` describes the Transparent Sensitive Data Protection policy and condition mapping. It also lists the property-value pairs for the condition.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `POLICY_NAME` | `VARCHAR2(128)` | | The name of the Transparent Sensitive Data Protection policy |
| `SUB_POLICY` | `NUMBER` | `NOT NULL` | The sub policy of the Transparent Sensitive Data Protection policy |
| `PROPERTY` | `VARCHAR2(11)` | | The condition property. Possible values:<br>• `DATATYPE`<br>• `LENGTH`<br>• `SCHEMA_NAME`<br>• `TABLE_NAME` |
| `VALUE` | `VARCHAR2(128)` | | The value of the condition property |

**See Also:**

*Oracle Database Security Guide* for more information about using Transparent Sensitive Data Protection

# DBA_TSDP_POLICY_FEATURE

`DBA_TSDP_POLICY_FEATURE` shows the Transparent Sensitive Data Protection policy security feature mapping for all the TSDP policies in the database. At this time, only Oracle Data Redaction is supported.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| `POLICY_NAME` | `VARCHAR2(128)` | | The name of the Transparent Sensitive Data Protection policy |
| `SECURITY_FEATURE` | `VARCHAR2(12)` | | The Oracle security feature with which the Transparent Sensitive Data Protection policy is associated |

**See Also:**

*Oracle Database Security Guide* for more information about using Transparent Sensitive Data Protection

# DBA_TSDP_POLICY_PARAMETER

DBA_TSDP_POLICY_PARAMETER shows the parameter-value pairs for the condition of the Transparent Sensitive Data Protection policy.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| POLICY_NAME | VARCHAR2(128) | | The name of the Transparent Sensitive Data Protection policy |
| SUB_POLICY | NUMBER | NOT NULL | The sub policy of the Transparent Sensitive Data Protection policy |
| PARAMETER | VARCHAR2(128) | | The parameter for the Transparent Sensitive Data Protection sub policy |
| VALUE | VARCHAR2(4000) | | The value of the parameter |
| DEFAULT_OPTION | VARCHAR2(5) | | Indicates whether this is the default option for the policy:<br>• TRUE: This is the default option for the policy<br>• FALSE: This is not the default option for the policy |

**See Also:**

*Oracle Database Security Guide* for more information about using Transparent Sensitive Data Protection

# DBA_TSDP_POLICY_PROTECTION

DBA_TSDP_POLICY_PROTECTION shows the list of columns that have been protected through Transparent Sensitive Data Protection.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEMA_NAME | VARCHAR2(128) | NOT NULL | The schema containing the sensitive data |
| TABLE_NAME | VARCHAR2(128) | NOT NULL | The table containing the sensitive column |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | The name of the sensitive column |
| TSDP_POLICY | VARCHAR2(128) | | The TSDP policy name based on which the column protection was enabled |
| SECURITY_FEATURE | VARCHAR2(12) | | The security feature enabled on the sensitive column |
| SECURITY_FEATURE_POLICY | VARCHAR2(128) | NOT NULL | Name of the underlying Oracle security feature policy |
| SUBPOLICY# | NUMBER | NOT NULL | The subpolicy of the Transparent Sensitive Data Protection policy based on which protection has been enabled |

**See Also:**

*Oracle Database Security Guide* for more information about using Transparent Sensitive Data Protection

## DBA_TSDP_POLICY_TYPE

DBA_TSDP_POLICY_TYPE shows the Transparent Sensitive Data Protection policy to sensitive column type mapping.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| POLICY_NAME | VARCHAR2(128) | | The Transparent Sensitive Data Protection policy name |
| SENSITIVE_TYPE | VARCHAR2(128) | | The sensitive column type name |

**See Also:**

*Oracle Database Security Guide* for more information about using Transparent Sensitive Data Protection

## DBA_TSM_DESTINATION

DBA_TSM_DESTINATION lists transparent session migration (TSM) destination session statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_DATABASE_NAME | VARCHAR2(4000) | | Database name of source session |
| DESTINATION_DATABASE_NAME | VARCHAR2(4000) | | Database name of destination session |
| DESTINATION_INSTANCE_NAME | VARCHAR2(4000) | | Instance name of destination session |
| DESTINATION_INSTANCE_ID | VARCHAR2(4000) | | Instance ID of destination session |
| DESTINATION_INST_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Instance start time of destination session |
| SEQUENCE# | NUMBER | | Migration sequence number |
| DESTINATION_SID | NUMBER | | Session ID of destination session |
| DESTINATION_SERIAL# | NUMBER | | Session serial number of destination session |
| DESTINATION_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Start time for migration on destination session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DESTINATION_END_TIME | TIMESTAMP(6) WITH TIME ZONE | | End time for migration on destination session |
| DESTINATION_USER_NAME | VARCHAR2(128) | NOT NULL | User associated with the destination session |
| DESTINATION_SCHEMA_NAME | VARCHAR2(128) | NOT NULL | Schema associated with the destination session |
| DESTINATION_STATE | VARCHAR2(24) | | Migration state of destination session |

# DBA_TSM_SOURCE

DBA_TSM_SOURCE lists transparent session migration (TSM) source session statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_DATABASE_NAME | VARCHAR2(4000) | | Database name of source session |
| SOURCE_INSTANCE_NAME | VARCHAR2(4000) | | Instance name of source session |
| SOURCE_INSTANCE_ID | VARCHAR2(4000) | | Instance ID of source session |
| SOURCE_INSTANCE_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Instance start time of source session |
| SEQUENCE# | NUMBER | | Migration sequence number |
| SOURCE_SID | NUMBER | | Session ID of source session |
| SOURCE_SERIAL# | NUMBER | | Source serial number of source session |
| SOURCE_STATE | VARCHAR2(24) | | Migration state of source session |
| CONNECT_STRING | VARCHAR2(4000) | | Connect string specified for migration |
| SOURCE_START_TIME | TIMESTAMP(6) WITH TIME ZONE | | Start time for migration on source session |
| COST | NUMBER | | Estimate of migration cost |
| FAILURE_REASON | VARCHAR2(34) | | Reason for migration failure, if any |
| SOURCE_END_TIME | TIMESTAMP(6) WITH TIME ZONE | | End time for migration on source session |
| ROUNDTRIPS | NUMBER | | Number of client/server round trips during migration |
| SOURCE_USER_NAME | VARCHAR2(128) | NOT NULL | User associated with the source session |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_SCHEMA_NAME | VARCHAR2(128) | NOT NULL | Schema associated with the source session |
| DESTINATION_DATABASE_NAME | VARCHAR2(4000) | | Database name of the destination session |

# DBA_TSTZ_TAB_COLS

DBA_TSTZ_TAB_COLS displays information about the columns of all tables in the database, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types. Its columns (except for COLUMN_NAME, NESTED and VIRTUAL_COLUMN) are the same as those in ALL_TSTZ_TAB_COLS.

**See Also:**

"ALL_TSTZ_TAB_COLS"

# DBA_TSTZ_TABLES

DBA_TSTZ_TABLES displays information about all tables in the database, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types. Its columns are the same as those in ALL_TSTZ_TABLES.

**See Also:**

"ALL_TSTZ_TABLES"

# DBA_TUNE_MVIEW

DBA_TUNE_MVIEW displays the result of executing the DBMS_ADVISOR.TUNE_MVIEW procedure.

### Related View

USER_TUNE_MVIEW displays the result of executing the DBMS_ADVISOR.TUNE_MVIEW procedure. This view does not display the OWNER column.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | | Owner of the task |
| TASK_NAME | VARCHAR2(128) | | Name of the task |
| ACTION_ID | NUMBER | NOT NULL | Identifier of the action |
| SCRIPT_TYPE | VARCHAR2(14) | | Type of the script:<br>• IMPLEMENTATION<br>• UNDO |
| STATEMENT | CLOB | | Action statement |

**See Also:**

- "USER_TUNE_MVIEW"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_ADVISOR.TUNE_MVIEW procedure

## DBA_TYPE_ATTRS

DBA_TYPE_ATTRS describes the attributes of all object types in the database. Its columns (except for CHAR_USED) are the same as those in ALL_TYPE_ATTRS.

**See Also:**

"ALL_TYPE_ATTRS"

## DBA_TYPE_METHODS

DBA_TYPE_METHODS describes the methods of all object types in the database. Its columns are the same as those in ALL_TYPE_METHODS.

**See Also:**

"ALL_TYPE_METHODS"

## DBA_TYPE_VERSIONS

DBA_TYPE_VERSIONS describes the versions of all object types in the database. Its columns are the same as those in ALL_TYPE_VERSIONS.

**See Also:**

"ALL_TYPE_VERSIONS"

## DBA_TYPES

DBA_TYPES describes all object types in the database. Its columns are the same as those in ALL_TYPES.

**See Also:**

"ALL_TYPES"

## DBA_UNDO_EXTENTS

DBA_UNDO_EXTENTS describes the extents comprising the segments in all undo tablespaces in the database.

**Note:**

The status of the undo space distribution reported by DBA_UNDO_EXTENTS is correct for the undo tablespace that is active on the instance on which DBA_UNDO_EXTENTS is queried. However, due to the use of in-memory information that is different on each instance, there can be a discrepancy in the status of the undo space distribution of undo tablespaces active on other instances when queried from one instance. This does not affect undo functionality and is only a reporting discrepancy for other instances' undo tablespace space distribution status. As a best practice, query the space distribution for an undo tablespace from the instance on which it is active.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | CHAR(3) | | Owner of the undo tablespace |
| SEGMENT_NAME | VARCHAR2(128) | NOT NULL | Name of the undo segment |
| TABLESPACE_NAME | VARCHAR2(128) | NOT NULL | Name of the undo tablespace |
| EXTENT_ID | NUMBER | | ID of the extent |
| FILE_ID | NUMBER | NOT NULL | File identifier number of the file containing the extent |
| BLOCK_ID | NUMBER | | Start block number of the extent |
| BYTES | NUMBER | | Size of the extent (in bytes) |
| BLOCKS | NUMBER | | Size of the extent (in blocks) |
| RELATIVE_FNO | NUMBER | | Relative number of the file containing the segment header |
| COMMIT_JTIME | NUMBER | | Commit time of the undo in the extent expressed as Julian time. This column is deprecated, but retained for backward compatibility reasons. |
| COMMIT_WTIME | VARCHAR2(20) | | Commit time of the undo in the extent expressed as Wallclock time.This column is deprecated, but retained for backward compatibility reasons. |
| STATUS | VARCHAR2(9) | | Transaction Status of the undo in the extent:<br>• ACTIVE<br>• EXPIRED<br>• UNEXPIRED |

# DBA_UNUSED_COL_TABS

DBA_UNUSED_COL_TABS describes all tables in the database containing unused columns. Its columns are the same as those in ALL_UNUSED_COL_TABS.

**See Also:**

"ALL_UNUSED_COL_TABS"

# DBA_UNUSED_OBJPRIVS

DBA_UNUSED_OBJPRIVS lists the object privileges (without privilege grant paths) that are not used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of the privilege analysis policy |
| USERNAME | VARCHAR2(128) | | Name of the user whose privileges are reported |
| ROLENAME | VARCHAR2(128) | | Name of the role whose unused privileges are reported (for ROLE type privilege analysis or ROLE AND CONTEXT privilege analysis) |
| OBJ_PRIV | VARCHAR2(40) | | Unused object privilege |
| OBJECT_OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object that USERNAME has OBJ_PRIV on |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object USERNAME has OBJ_PRIV on |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column that USERNAME has OBJ_PRIV on |
| GRANT_OPTION | NUMBER | | Indicates whether the privilege is granted with the GRANT option:<br>• 0 - Indicates that the privilege is granted without the GRANT option<br>• 1 - Indicates that the privilege is granted with the GRANT option |

**See Also:**

- "DBA_UNUSED_OBJPRIVS_PATH" for privilege grant path information for unused object privileges
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_UNUSED_OBJPRIVS_PATH

DBA_UNUSED_OBJPRIVS_PATH lists the object privileges that are not used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of the privilege analysis policy |
| USERNAME | VARCHAR2(128) | | Name of the user whose privileges are reported |
| ROLENAME | VARCHAR2(128) | | Name of the role whose unused privileges are reported (for ROLE type privilege analysis or ROLE AND CONTEXT privilege analysis) |
| OBJ_PRIV | VARCHAR2(40) | | Unused object privilege |
| OBJECT_OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object that USERNAME has OBJ_PRIV on |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object that USERNAME has OBJ_PRIV on |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column that USERNAME has OBJ_PRIV on |
| GRANT_OPTION | NUMBER | | Indicates whether the privilege is granted with the GRANT option:<br>• 0 - Indicates that the privilege is granted without the GRANT option<br>• 1 - Indicates that the privilege is granted with the GRANT option |
| PATH | GRANT_PATH | | Object privilege grant paths |

**See Also:**

- "DBA_USED_OBJPRIVS_PATH"
- "DBA_UNUSED_OBJPRIVS"
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_UNUSED_PRIVS

DBA_UNUSED_PRIVS lists the privileges that are not used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of the privilege analysis policy |
| USERNAME | VARCHAR2(128) | | Name of the user whose unused privileges are reported |
| ROLENAME | VARCHAR2(128) | | Name of the role whose unused privileges are reported (for ROLE type privilege analysis or ROLE AND CONTEXT privilege analysis) |
| SYS_PRIV | VARCHAR2(40) | | Unused system privilege |
| OBJ_PRIV | VARCHAR2(40) | | Unused object privilege |
| USER_PRIV | VARCHAR2(18) | | Unused user privilege |
| OBJECT_OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object that USERNAME has OBJ_PRIV or USER_PRIV on |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object that OBJ_PRIV has accessed or USER if USER_PRIV was used |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column that OBJ_PRIV has access on |
| OPTION$ | NUMBER | | Indicates whether the privilege is granted with the GRANT option or the ADMIN option:<br>• 0 - Indicates that the privilege is granted without the GRANT option or ADMIN option<br>• 1 - Indicates that the privilege is granted with the GRANT option or ADMIN option |
| PATH | GRANT_PATH | | Privilege grant paths |

**See Also:**

- "DBA_USED_PRIVS"
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_UNUSED_SYSPRIVS

DBA_UNUSED_SYSPRIVS lists the system privileges (without privilege grant paths) that are not used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| USERNAME | VARCHAR2(128) | | Name of the user whose privileges are reported |
| ROLENAME | VARCHAR2(128) | | Name of the role whose unused privileges are reported (for ROLE type privilege analysis or ROLE AND CONTEXT privilege analysis) |
| SYS_PRIV | VARCHAR2(40) | | Unused system privilege |
| ADMIN_OPTION | NUMBER | | Indicates whether the privilege is granted with the ADMIN option:<br>• 0 - Indicates that the privilege is granted without the ADMIN option<br>• 1 - Indicates that the privilege is granted with the ADMIN option |

**See Also:**

- "DBA_UNUSED_SYSPRIVS_PATH" for privilege grant path information for unused system privileges
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_UNUSED_SYSPRIVS_PATH

DBA_UNUSED_SYSPRIVS_PATH lists the system privileges that are not used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| USERNAME | VARCHAR2(128) | | Name of the user whose privileges are reported |
| ROLENAME | VARCHAR2(128) | | Name of the role whose unused privileges are reported (for ROLE type privilege analysis or ROLE AND CONTEXT privilege analysis) |
| SYS_PRIV | VARCHAR2(40) | | Unused system privilege |
| ADMIN_OPTION | NUMBER | | Indicates whether the privilege is granted with the ADMIN option:<br>• 0 - Indicates that the privilege is granted without the ADMIN option<br>• 1 - Indicates that the privilege is granted with the ADMIN option |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PATH | GRANT_PATH | | System privilege grant paths |

**See Also:**

- "DBA_USED_SYSPRIVS_PATH"
- "DBA_UNUSED_SYSPRIVS"
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_UNUSED_USERPRIVS

DBA_UNUSED_USERPRIVS lists the user privileges (without privilege grant paths) that are not used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| USERNAME | VARCHAR2(128) | | Name of the user whose privileges are reported |
| ROLENAME | VARCHAR2(128) | | Name of the role whose unused privileges are reported (for ROLE type privilege analysis or ROLE AND CONTEXT privilege analysis) |
| USER_PRIV | VARCHAR2(18) | | Unused user privilege |
| ONUSER | VARCHAR2(128) | | The user whose user privileges the grantee can exercise |
| GRANT_OPTION | NUMBER | | Indicates whether the privilege is granted with the GRANT option:<br>• 0 - Indicates that the privilege is granted without the GRANT option<br>• 1 - Indicates that the privilege is granted with the GRANT option |

**See Also:**

- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_UNUSED_USERPRIVS_PATH

DBA_UNUSED_USERPRIVS_PATH lists the user privileges that are not used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| USERNAME | VARCHAR2(128) | | Name of the user whose privileges are reported |
| ROLENAME | VARCHAR2(128) | | Name of the role whose unused privileges are reported (for ROLE type privilege analysis or ROLE AND CONTEXT privilege analysis) |
| USER_PRIV | VARCHAR2(18) | | Unused user privilege |
| ONUSER | VARCHAR2(128) | | The user whose user privileges the grantee can exercise |
| GRANT_OPTION | NUMBER | | Indicates whether the privilege is granted with the GRANT option:<br>• 0 - Indicates that the privilege is granted without the GRANT option<br>• 1 - Indicates that the privilege is granted with the GRANT option |
| PATH | GRANT_PATH | | User privilege grant paths |

**See Also:**

- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_UPDATABLE_COLUMNS

DBA_UPDATABLE_COLUMNS describes all columns in a join view that can be updated by the database administrator, subject to appropriate privileges. Its columns are the same as those in "ALL_UPDATABLE_COLUMNS".

**See Also:**

*Oracle Database Concepts* for information on updatable join views

# DBA_USED_OBJPRIVS

DBA_USED_OBJPRIVS lists the object privileges (without privilege grant paths) that are used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of the privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was reported |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privilege was reported |
| USED_ROLE | VARCHAR2(128) | | Used role |
| OBJ_PRIV | VARCHAR2(40) | | Used object privilege |
| OBJECT_OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object that OBJ_PRIV was used to access |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object that OBJ_PRIV was used to access |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column that OBJ_PRIV was used to access |
| GRANT_OPTION | NUMBER | | Indicates whether the GRANT option was used:<br>• 0 - Indicates that the GRANT option was not used<br>• 1 - Indicates that the GRANT option was used |

**See Also:**

- "DBA_USED_OBJPRIVS_PATH" for privilege grant path information for used object privileges
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USED_OBJPRIVS_PATH

DBA_USED_OBJPRIVS_PATH lists the object privileges that are used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of the privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was reported |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privilege was reported |
| USED_ROLE | VARCHAR2(128) | | Used role |
| OBJ_PRIV | VARCHAR2(40) | | Used object privilege |
| OBJECT_OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object that OBJ_PRIV is used to access |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object that OBJ_PRIV is used to access |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column that OBJ_PRIV is used to access |
| GRANT_OPTION | NUMBER | | Whether the GRANT option of the privilege is used |
| PATH | GRANT_PATH | | Object privilege grant paths |

**See Also:**

- "DBA_UNUSED_OBJPRIVS_PATH"
- "DBA_USED_OBJPRIVS"
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USED_PRIVS

DBA_USED_PRIVS lists the privileges that are used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was reported |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privilege was reported |
| USED_ROLE | VARCHAR2(128) | | Used role |
| SYS_PRIV | VARCHAR2(40) | | Used system privilege |
| OBJ_PRIV | VARCHAR2(40) | | Used object privilege |
| USER_PRIV | VARCHAR2(18) | | Used user privilege |
| OBJECT_OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object or user that OBJ_PRIV or USER_PRIV is used to access |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object or user that OBJ_PRIV or USER_PRIV is used to access |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column that OBJ_PRIV is used to access |
| OPTION$ | NUMBER | | Indicates whether the GRANT option or the ADMIN option was used:<br>• 0 - Indicates that the GRANT option or ADMIN option was not used<br>• 1 - Indicates that the GRANT option or ADMIN option was used |
| PATH | GRANT_PATH | | Used privilege grant paths |

**See Also:**

- "DBA_UNUSED_PRIVS"
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USED_PUBPRIVS

DBA_USED_PUBPRIVS lists the privileges that are used from the PUBLIC role for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was reported |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user who used the privilege from the PUBLIC role |
| SYS_PRIV | VARCHAR2(40) | | Used system privilege |
| OBJ_PRIV | VARCHAR2(40) | | Used object privilege |
| OBJECT_OWNER | VARCHAR2(128) | | Object owner |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object that OBJ_PRIV is used to access |
| OBJECT_TYPE | VARCHAR2(23) | | Type of the object that OBJ_PRIV is used to access |
| OPTION$ | NUMBER | | Indicates whether the GRANT option or the ADMIN option was used:<br>• 0 - Indicates that the GRANT option or ADMIN option was not used<br>• 1 - Indicates that the GRANT option or ADMIN option was used |

**See Also:**

- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USED_SYSPRIVS

DBA_USED_SYSPRIVS lists the system privileges (without privilege grant paths) that are used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was used |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privilege was reported |
| USED_ROLE | VARCHAR2(128) | | Used role |
| SYS_PRIV | VARCHAR2(40) | | Used system privilege |
| ADMIN_OPTION | NUMBER | | Indicates whether the ADMIN option was used:<br>• 0 - Indicates that the ADMIN option was not used<br>• 1 - Indicates that the ADMIN option was used |

**See Also:**

- "DBA_USED_SYSPRIVS_PATH" for privilege grant path information for used system privileges
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USED_SYSPRIVS_PATH

DBA_USED_SYSPRIVS_PATH lists the system privileges that are used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was reported |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privilege was reported |
| USED_ROLE | VARCHAR2(128) | | Used role |
| SYS_PRIV | VARCHAR2(40) | | Used system privilege |
| ADMIN_OPTION | NUMBER | | Indicates whether the ADMIN option was used:<br>• 0 - Indicates that the ADMIN option was not used<br>• 1 - Indicates that the ADMIN option was used |
| PATH | GRANT_PATH | | System privilege grant paths |

**See Also:**

- "DBA_UNUSED_SYSPRIVS_PATH"
- "DBA_USED_SYSPRIVS"
- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USED_USERPRIVS

DBA_USED_USERPRIVS lists the user privileges (without privilege grant paths) that are used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was reported |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privilege was reported |
| USED_ROLE | VARCHAR2(128) | | Used role |
| USER_PRIV | VARCHAR2(18) | | Used user privilege |
| ONUSER | VARCHAR2(128) | | The user whose user privileges the grantee can exercise |
| GRANT_OPTION | NUMBER | | Indicates whether the privilege is granted with the GRANT option:<br>• 0 - Indicates that the privilege is granted without the GRANT option<br>• 1 - Indicates that the privilege is granted with the GRANT option |

**See Also:**

- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USED_USERPRIVS_PATH

DBA_USED_USERPRIVS_PATH lists the user privileges that are used for the privilege analysis policies reported by the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure.

This view provides access to analyzed privilege records in SYS tables.

You must have the CAPTURE_ADMIN role to access this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE | VARCHAR2(128) | NOT NULL | Name of a privilege analysis policy |
| SEQUENCE | NUMBER | NOT NULL | The sequence number of the privilege analysis run during which the privilege was reported |
| OS_USER | VARCHAR2(128) | | Operating system login username |
| USERHOST | VARCHAR2(128) | | Client host machine name |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MODULE | VARCHAR2(64) | | Module name |
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user whose privilege was reported |
| USED_ROLE | VARCHAR2(128) | | Used role |
| USER_PRIV | VARCHAR2(18) | | Used user privilege |
| ONUSER | VARCHAR2(128) | | The user whose user privileges the grantee can exercise |
| GRANT_OPTION | NUMBER | | Indicates whether the privilege is granted with the GRANT option:<br>• 0 - Indicates that the privilege is granted without the GRANT option<br>• 1 - Indicates that the privilege is granted with the GRANT option |
| PATH | GRANT_PATH | | User privilege grant paths |

**See Also:**

- *Oracle Database Vault Administrator's Guide* for more information about privilege analysis
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure

# DBA_USERS

DBA_USERS describes all users of the database.

### Related View

USER_USERS describes the current user. This view does not display the PASSWORD, PROFILE, PASSWORD_VERSIONS, EDITIONS_ENABLED, AUTHENTICATION_TYPE , and LAST_LOGIN columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user |
| USER_ID | NUMBER | NOT NULL | ID number of the user |
| PASSWORD | VARCHAR2(4000) | | This column is deprecated in favor of the AUTHENTICATION_TYPE column |
| ACCOUNT_STATUS | VARCHAR2(32) | NOT NULL | Account status:<br>• OPEN<br>• EXPIRED<br>• EXPIRED(GRACE)<br>• LOCKED(TIMED)<br>• LOCKED<br>• EXPIRED & LOCKED(TIMED) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • EXPIRED(GRACE) & LOCKED(TIMED)<br>• EXPIRED & LOCKED<br>• EXPIRED(GRACE) & LOCKED |
| LOCK_DATE | DATE | | Date the account was locked if account status was LOCKED |
| EXPIRY_DATE | DATE | | Date of expiration of the account |
| DEFAULT_TABLESPACE | VARCHAR2(30) | NOT NULL | Default tablespace for data |
| TEMPORARY_TABLESPACE | VARCHAR2(30) | NOT NULL | Name of the default tablespace for temporary tables or the name of a tablespace group |
| CREATED | DATE | NOT NULL | User creation date |
| PROFILE | VARCHAR2(128) | NOT NULL | User resource profile name |
| INITIAL_RSRC_CONSUMER_GROUP | VARCHAR2(128) | | Initial resource consumer group for the user |
| EXTERNAL_NAME | VARCHAR2(4000) | | User external name |
| PASSWORD_VERSIONS | VARCHAR2(12) | | Shows the list of versions of the password hashes (also known as "verifiers") existing for the account.<br>The PASSWORD_VERSIONS column value includes 10G if an old case-insensitive ORCL hash exists, 11G if a SHA-1 hash exists, and 12C if a de-optimized PBKDF2-based hash exists.<br>For more information about the 12C verifier, see *Oracle Database Security Guide*.<br>Note that any combination of these verifiers can exist for any given account. |
| EDITIONS_ENABLED | VARCHAR2(1) | | Indicates whether editions have been enabled for the corresponding user (Y) or not (N) |
| AUTHENTICATION_TYPE | VARCHAR2(8) | | Indicates the authentication mechanism for the user:<br>• EXTERNAL - CREATE USER *user1* IDENTIFIED EXTERNALLY;<br>• GLOBAL - CREATE USER *user2* IDENTIFIED GLOBALLY;<br>• PASSWORD - CREATE USER *user3* IDENTIFIED BY *user3*; |
| PROXY_ONLY_CONNECT | VARCHAR2(1) | | Indicates whether a user can connect directly (N) or whether the account can only be proxied (Y) by users who have proxy privileges for this account (that is, by users who have been granted the "connect through" privilege for this account).<br>For more information about creating proxy user accounts and authorizing users to connect through them, see *Oracle Database Security Guide*. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMMON | VARCHAR2(3) | | Indicates whether a given user is common. Possible values<br>• YES if a user is common<br>• NO if a user is local (not common) |
| LAST_LOGIN | TIMESTAMP(9) WITH TIME ZONE | | The time of the last user login. |
| ORACLE_MAINTAINED | VARCHAR2(1) | | Denotes whether the user was created, and is maintained, by Oracle-supplied scripts (such as catalog.sql or catproc.sql). A user for which this column has the value Y must not be changed in any way except by running an Oracle-supplied script. |

**See Also:**

"USER_USERS"

# DBA_USERS_WITH_DEFPWD

DBA_USERS_WITH_DEFPWD displays all users in the database that are still using their default passwords.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user |
| PRODUCT | VARCHAR2(4000) | | Name of the product the user belongs to |

**Note:**

In a CDB, when DBA_USERS_WITH_DEFPWD is queried from a PDB, information about local users who are using their default passwords is displayed. To display information about common users, query DBA_USERS_WITH_DEFPWD from the root.

**See Also:**

*Oracle Database Concepts* for an introduction to local and common users in a CDB

# DBA_USTATS

DBA_USTATS describes the user-defined statistics collected on all tables and indexes in the database. Its columns are the same as those in ALL_USTATS.

**See Also:**

"ALL_USTATS"

## DBA_VARRAYS

DBA_VARRAYS describes all varrays in the database. Its columns are the same as those in ALL_VARRAYS.

**See Also:**

"ALL_VARRAYS"

## DBA_VIEWS

DBA_VIEWS describes all views in the database. Its columns are the same as those in ALL_VIEWS.

**See Also:**

"ALL_VIEWS"

## DBA_VIEWS_AE

DBA_VIEWS_AE describes all views (across all editions) in the database. Its columns are the same as those in ALL_VIEWS_AE.

**See Also:**

"ALL_VIEWS_AE"

## DBA_WAITERS

DBA_WAITERS shows all the sessions that are waiting for a lock. In an Oracle RAC environment, this only applies if the waiter is on the same instance.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WAITING_SESSION | NUMBER | | The waiting session |
| WAITING_CON_ID | NUMBER | | The container ID (CON_ID) of the waiting session |
| HOLDING_SESSION | NUMBER | | The holding session |
| HOLDING_CON_ID | NUMBER | | The container ID (CON_ID) of the holding session. |
| LOCK_TYPE | VARCHAR2(26) | | The lock type |
| MODE_HELD | VARCHAR2(40) | | The mode held |
| MODE_REQUESTED | VARCHAR2(40) | | The mode requested |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| LOCK_ID1 | NUMBER | | Lock ID 1 |
| LOCK_ID2 | NUMBER | | Lock ID 2 |

# DBA_WALLET_ACES

DBA_WALLET_ACES describes access control entries defined in wallet access control lists.

### Related View

USER_WALLET_ACES describes the status of access control entries for the current user to access wallets through PL/SQL network utility packages. This view does not display the ACE_ORDER, START_DATE, END_DATE, GRANT_TYPE, INVERTED_PRINCIPAL, PRINCIPAL, or PRINCIPAL_TYPE columns.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WALLET_PATH | VARCHAR2(1000) | NOT NULL | Wallet path |
| ACE_ORDER | NUMBER | NOT NULL | Order number of the access control entry |
| START_DATE | TIMESTAMP(6) | | Start date of the access control entry |
| END_DATE | TIMESTAMP(6) | | End date of the access control entry |
| GRANT_TYPE | VARCHAR2(5) | | Indicates whether the access control entry grants or denies the privilege |
| INVERTED_PRINCIPAL | VARCHAR2(3) | | Indicates whether the principal is inverted or not |
| PRINCIPAL | VARCHAR2(128) | | Principal the privilege is applied to |
| PRINCIPAL_TYPE | VARCHAR2(16) | | Type of the principal |
| PRIVILEGE | VARCHAR2(128) | | Privilege |

**See Also:**

"USER_WALLET_ACES"

# DBA_WALLET_ACLS

DBA_WALLET_ACLS displays the access control lists assigned to restrict access to wallets through PL/SQL network utility packages.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WALLET_PATH | VARCHAR2(1000) | NOT NULL | Wallet path |
| ACL | VARCHAR2(4000) | | Path of the access control list |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ACLID | RAW(8) | | Object ID of the access control list |
| ACL_OWNER | VARCHAR2(128) | | Owner of the access control list |

# DBA_WARNING_SETTINGS

DBA_WARNING_SETTINGS displays information about the warning parameter settings for all objects in the database. Its columns are the same as those in ALL_WARNING_SETTINGS.

**See Also:**

"ALL_WARNING_SETTINGS"

# DBA_WI_CAPTURE_FILES

Each row in DBA_WI_CAPTURE_FILES represents a capture file that belongs to the workload analyzed in the current Workload Intelligence job.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The identifier of the job in the workload to which the given capture file belongs |
| FILE_ID | NUMBER | NOT NULL | The identifier of the current capture file |
| FILE_PATH | VARCHAR2(4000) | NOT NULL | The path of the current capture file |

# DBA_WI_JOBS

Each row in DBA_WI_JOBS describes a Workload Intelligence job, that is, a task that applies the algorithms of Workload Intelligence on a given capture directory.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The job identifier |
| JOB_NAME | VARCHAR2(128) | NOT NULL | A name that uniquely identifies the given job |
| CAPTURE_DIRECTORY | VARCHAR2(4000) | NOT NULL | Path to the capture directory on which the given job has been applied |
| MODEL_ORDER | NUMBER | | The order of the markov model that describes the workload associated with the current job. If NULL, the corresponding order has not been calculated yet. |
| THRESHOLD | NUMBER | | A number in the range [0, 1] that represents the threshold that the user has given as an input parameter to the current job of Workload Intelligence for the identification of significant patterns. If NULL, the process of pattern identification has not been initiated yet. |

# DBA_WI_OBJECTS

Each row in DBA_WI_OBJECTS represents a database object (table) that is accessed by the given template in the given Workload Intelligence job.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The identifier of the job in the workload of which the given object has been accessed |
| TEMPLATE_ID | NUMBER | NOT NULL | The identifier of the template in the given job by which the current object has been accessed |
| OBJECT_ID | NUMBER | NOT NULL | The identifier of the current object |
| ACCESS_TYPE | VARCHAR2(2) | NOT NULL | Possible values:<br>• R - Indicates that the current object has been accessed for reading by the given template<br>• W - Indicates that the current object has been accessed for writing by the given template<br>• RW - Indicates that the current object has been accessed for both reading and writing by the given template |

# DBA_WI_PATTERN_ITEMS

Each row in DBA_WI_PATTERN_ITEMS represents a template that participates in a significant pattern that has been found by the given Workload Intelligence job.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The identifier of the job in the workload of which the current pattern has been found |
| PATTERN_ID | NUMBER | NOT NULL | The identifier of the pattern to which the current item (template) belongs |
| SEQUENCE_NUMBER | NUMBER | NOT NULL | Number that indicates the position of the current item in the given pattern |
| TEMPLATE_ID | NUMBER | NOT NULL | The identifier of the template that participates in the given position of the current pattern |
| IS_FIRST_IN_LOOP | CHAR(1) | NOT NULL | A flag that indicates whether or not the current item marks the beginning of a loop in the given pattern. The possible values are Y and N. |
| IS_LAST_IN_LOOP | CHAR(1) | NOT NULL | A flag that indicates whether or not the current item marks the end of a loop in the given pattern. The possible values are Y and N. |

# DBA_WI_PATTERNS

Each row in DBA_WI_PATTERNS represents a pattern that has been identified by Workload Intelligence as significant in the workload associated with the given job. Such a pattern consists of one or more templates. These templates that comprise the given pattern are described in the related view DBA_WI_PATTERN_ITEMS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The identifier of the job in the workload of which the current pattern has been found |
| PATTERN_ID | NUMBER | NOT NULL | The identifier of the current pattern |
| LENGTH | NUMBER | NOT NULL | The length of the pattern, that is, the number of items (templates) it consists of |
| NUMBER_OF_EXECUTIONS | NUMBER | NOT NULL | The number of times the current pattern has been executed in the given workload |
| DB_TIME | NUMBER | NOT NULL | The total time consumed in the database server by all the executions of the current pattern in the given workload |

**See Also:**

"DBA_WI_PATTERN_ITEMS"

# DBA_WI_STATEMENTS

Each row in DBA_WI_STATEMENTS describes a statement (SQL or PL/SQL) that is part of the template with identifier TEMPLATE_ID, which has been found in the workload that is related to the Workload Intelligence job whose identifier is equal to JOB_ID. A template may consist of multiple statements, for example, if it represents a transaction. In this case, there is one row in this view for every one of these statements. These statements are ordered, based on the order defined by the corresponding transaction. Column SEQUENCE_NUMBER is used to describe this order.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The identifier of the job in the workload of which the given statement has been found |
| TEMPLATE_ID | NUMBER | NOT NULL | The identifier of the template in the given job to which the current statement belongs |
| SEQUENCE_NUMBER | NUMBER | NOT NULL | A number that indicates the order of the current statement in the given template |
| SQL_TEXT | CLOB | NOT NULL | The SQL text associated with the current statement. Note that although multiple SQL statements can be classified to the same template, only one row is stored that represents them all. This row corresponds to the first instance of the given template that is found during parsing of the workload. |

# DBA_WI_TEMPLATE_EXECUTIONS

Each row in DBA_WI_TEMPLATE_EXECUTIONS represents an execution of a template in a capture that belongs to the workload that is associated with the current Workload Intelligence job.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The identifier of the job in the workload of which the current execution of the given template belongs |
| CAPTURE_FILE_ID | NUMBER | NOT NULL | The identifier of the capture file in which the current execution of the given template was found |
| SEQUENCE_NUMBER | NUMBER | NOT NULL | A number that indicates the order of the current execution in the given capture file |
| TEMPLATE_ID | NUMBER | NOT NULL | The identifier of the template that was executed in the execution represented by the current row |
| DB_TIME | NUMBER | NOT NULL | The time that the current execution consumed on the database server |

# DBA_WI_TEMPLATES

Each row in DBA_WI_TEMPLATES describes a template that has been found in the workload that is related to the Workload Intelligence job whose identifier is equal to JOB_ID. A template can represent either a simple query, or an entire transaction. Two queries in the given workload belong to the same template, if they exhibit trivial differences, for example, if they contain different literal values, different bind variable names, different comments, or different white spaces.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| JOB_ID | NUMBER | NOT NULL | The identifier of the job in the workload of which the given template has been found |
| TEMPLATE_ID | NUMBER | NOT NULL | The identifier of a template in a given job |
| IS_TRANSACTION | CHAR(1) | NOT NULL | Flag that indicates whether the given template represents a transaction:<br>• Y - indicates that the given template represents a transaction<br>• N - indicates that the given template does not represent a transaction |

# DBA_WORKLOAD_ACTIVE_USER_MAP

DBA_WORKLOAD_ACTIVE_USER_MAP contains the mappings that are going to be valid for the next replay or are valid for the current replay.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEDULE_CAP_ID | NUMBER | | The ID of a capture in the schedule |
| CAPTURE_USER | VARCHAR2(4000) | NOT NULL | The user name during the time of the workload capture |
| REPLAY_USER | VARCHAR2(4000) | | The user name to which captured user should be remapped during replay |

# DBA_WORKLOAD_CAPTURES

DBA_WORKLOAD_CAPTURES displays all the workload captures that have been performed in the current database. It also lists captures on which DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFO() or DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO() have been called. Each row contains information about one workload capture.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ID | NUMBER | NOT NULL | Internal key for the workload capture |
| NAME | VARCHAR2(128) | NOT NULL | Name for the workload capture |
| DBID | NUMBER | NOT NULL | ID of the database in which the workload was captured |
| DBNAME | VARCHAR2(10) | NOT NULL | Name of the database in which the workload was captured |
| DBVERSION | VARCHAR2(128) | NOT NULL | Version of the database in which the workload was captured |
| PARALLEL | VARCHAR2(3) | | Indicates whether the database in which the workload was captured is an Oracle RAC database (YES) or a single instance database (NO) |
| DIRECTORY | VARCHAR2(128) | NOT NULL | Name of the directory object for workload capture |
| STATUS | VARCHAR2(40) | NOT NULL | Current status of the workload capture:<br>• IN PROGRESS - Workload capture is in progress<br>• COMPLETED - Workload capture has completed successfully<br>• FAILED - Workload capture was aborted due to errors encountered |
| START_TIME | DATE | NOT NULL | Datetime when the capture began |
| END_TIME | DATE | | Datetime when the capture completed or failed; NULL if the capture is still in progress |
| DURATION_SECS | NUMBER | | Duration of the workload capture (in seconds) |
| START_SCN | NUMBER | NOT NULL | Start SCN value for this capture |
| END_SCN | NUMBER | | End SCN value for this capture; NULL if the capture is still in progress |
| DEFAULT_ACTION | VARCHAR2(30) | NOT NULL | Mode in which to apply workload capture filters:<br>• INCLUDE - All the capture filters are treated as EXCLUSION filters, determining the workload that will not be captured.<br>• EXCLUDE - All the capture filters are treated as INCLUSION FILTERS, determining the workload that will be captured. |
| FILTERS_USED | NUMBER | | Number of filters that were used for this capture |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAPTURE_SIZE | NUMBER | | Total size of workload capture |
| DBTIME | NUMBER | | Total amount of database time (in microseconds) that has been recorded in this workload capture |
| DBTIME_TOTAL | NUMBER | | Total amount of database time (in microseconds) across the entire database during the workload capture, including the part of the workload that was not captured. |
| USER_CALLS | NUMBER | | Total number of user calls that have been recorded in this workload capture |
| USER_CALLS_TOTAL | NUMBER | | Total number of user calls across the entire database during the workload capture, including the part of the workload that was not captured. |
| USER_CALLS_UNREPLAYABLE | NUMBER | | Total number of user calls that will not be replayed in a subsequent replay of this workload capture |
| TRANSACTIONS | NUMBER | | Total number of transactions that have been recorded in this workload capture |
| TRANSACTIONS_TOTAL | NUMBER | | Total number of transactions across the entire database during the workload capture, including the part of the workload that was not captured. |
| CONNECTS | NUMBER | | Total number of session connects that have been recorded in this workload capture |
| CONNECTS_TOTAL | NUMBER | | Total number of session connects across the entire database during the workload capture, including the part of the workload that was not captured |
| ERRORS | NUMBER | | Total number of errors that have been recorded in this workload capture |
| AWR_DBID | NUMBER | | Database ID of the AWR snapshots that correspond to this workload capture. For captures that were performed in the current database, this value is equal to the current database's DBID. For captures that were performed in other databases, this value will either be NULL or will be populated by `DBMS_WORKLOAD_CAPTURE.IMPORT_AWR()`. |
| AWR_BEGIN_SNAP | NUMBER | | Begin snapshot ID of the AWR snapshots that correspond to this workload capture |
| AWR_END_SNAP | NUMBER | | End snapshot ID of the AWR snapshots that correspond to this workload capture |
| AWR_EXPORTED | VARCHAR2(12) | | Indicates whether the AWR snapshots that correspond to this workload capture have been exported using `DBMS_WORKLOAD_CAPTURE.EXPORT_AWR()` (`YES`) or not (`NO`), or whether AWR snapshots cannot be exported because the capture is still in progress, has run to completion successfully, or |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | was done in a different database from which it was not exported (NOT POSSIBLE) |
| ERROR_CODE | NUMBER | | Error code for this workload capture |
| ERROR_MESSAGE | VARCHAR2(300) | | Error message for this workload capture |
| DIR_PATH | VARCHAR2(4000) | NOT NULL | Full directory path for the workload capture directory object |
| DIR_PATH_SHARED | VARCHAR2(10) | NOT NULL | Indicates whether the workload capture directory is shared by all the instances of the recording database (applicable only for Oracle RAC databases) |
| LAST_PROCESSED_VERSION | VARCHAR2(128) | | Database version in which this capture was preprocessed using DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE() last; NULL if the capture has never been preprocessed |
| SQLSET_OWNER | VARCHAR2(128) | | User name of the SQL tuning set owner |
| SQLSET_NAME | VARCHAR2(128) | | Name of the SQL tuning set for this workload capture |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_WORKLOAD_CAPTURE package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_WORKLOAD_REPLAY package

# DBA_WORKLOAD_CONNECTION_MAP

DBA_WORKLOAD_CONNECTION_MAP displays the connection mapping information for workload replay. Each row defines one connection mapping for a particular workload replay.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REPLAY_ID | NUMBER | NOT NULL | ID of the replay (corresponds to DBA_WORKLOAD_REPLAYS.ID) |
| CONN_ID | NUMBER | NOT NULL | Key (ID) of the connection mapping table |
| SCHEDULE_CAP_ID | NUMBER | | Schedule capture ID (corresponds to DBA_WORKLOAD_SCHEDULE_CAPTURES.SCHEDULE_CAP_ID) |
| CAPTURE_CONN | VARCHAR2(4000) | NOT NULL | Connection string that was used during capture |
| REPLAY_CONN | VARCHAR2(4000) | | Connection string that should be used during replay |

# DBA_WORKLOAD_FILTERS

DBA_WORKLOAD_FILTERS displays all the workload filters that have been defined in the current database. In Oracle Database 11*g*, only workload filters of type CAPTURE are supported. Starting with Oracle Database 11*g*R2, filters of type REPLAY are supported.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE | VARCHAR2(30) | | Type of the workload filter (CAPTURE or REPLAY) |
| ID | VARCHAR2(40) | | Sequence number of the workload filter |
| STATUS | VARCHAR2(6) | | Status of the workload filter:<br>• NEW - This filter will be used by the next subsequent operation such as the next workload capture.<br>• IN USE - This filter is currently being used by an operation that is in progress such as an active workload capture.<br>• USED - This filter was used in the past by some operation such as a past workload capture. |
| SET_NAME | VARCHAR2(1000) | | Name of the filter set to which the filter belongs |
| NAME | VARCHAR2(128) | | Name of the workload filter |
| ATTRIBUTE | VARCHAR2(128) | | Name of the attribute on which the filter is defined |
| VALUE | VARCHAR2(4000) | | Value of the attribute on which the filter is defined. Wildcards such as % and _ are supported if the attribute is of string type. |

# DBA_WORKLOAD_REPLAY_DIVERGENCE

DBA_WORKLOAD_REPLAY_DIVERGENCE displays information about data/error divergence for a user call that has been replayed.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REPLAY_ID | NUMBER | NOT NULL | ID (key) for the workload replay |
| TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Time that the divergence occurred |
| DIVERGENCE_TYPE | NUMBER | NOT NULL | Reserved for future use |
| IS_QUERY_DATA_DIVERGENCE | VARCHAR2(1) | | Indicates whether the data divergence is from the number of rows fetched by SELECT queries (Y) or not (N) |
| IS_DML_DATA_DIVERGENCE | VARCHAR2(1) | | Indicates whether the divergence is from the number of rows affected by INSERT, UPDATE, or DELETE SQL statements (Y) or not (N) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| IS_ERROR_DIVERGENCE | VARCHAR2(1) | | Indicates whether the divergence is from errors seen during capture or replay (Y) or not (N) |
| IS_THREAD_FAILURE | VARCHAR2(1) | | Indicates whether the divergence is from sessions that failed during replay (Y) or not (N) |
| IS_DATA_MASKED | VARCHAR2(1) | | Indicates whether the SQL call contains masked bind data (Y) or not (N).<br>If data masking technology is used at the replay database, the workload capture files need to be masked. Otherwise, SQL statements generated from capture files that contain sensitive bind data will not match the database. When the replay client sends masked bind data to the server, it turns on the IS_DATA_MASKED flag for the current SQL call. |
| EXPECTED_ROW_COUNT | NUMBER | | Number of rows fetched for SELECT queries or rows affected for INSERT, UPDATE, or DELETE SQL statements during capture |
| OBSERVED_ROW_COUNT | NUMBER | | Actual number of rows fetched for SELECT queries or rows affected for INSERT, UPDATE, or DELETE SQL statements during replay |
| EXPECTED_ERROR# | NUMBER | | Error number that was seen during capture (0 if the capture ran successfully) |
| EXPECTED_ERROR_MESSAGE | VARCHAR2(4000) | | Text of the error message whose number appears in the EXPECTED_ERROR# column |
| OBSERVED_ERROR# | NUMBER | | Actual error number seen during replay (0 if the replay ran successfully, 15566 (corresponding to ORA-15566) if the captured call could not be replayed) |
| OBSERVED_ERROR_MESSAGE | VARCHAR2(4000) | | Text of the error message whose number appears in the OBSERVED_ERROR# column |
| STREAM_ID | NUMBER | NOT NULL | Stream ID of the session that reported the divergence |
| CALL_COUNTER | NUMBER | NOT NULL | Call counter of the user call that reported the divergence |
| CAPTURE_STREAM_ID | NUMBER | | Internal ID of the capture file whose replay produced the divergence |
| SQL_ID | VARCHAR2(13) | | SQL ID of the SQL that reported the divergence |
| SESSION_ID | NUMBER | NOT NULL | Session ID of the session that reported the divergence |
| SESSION_SERIAL# | NUMBER | NOT NULL | Captured session serial number of the session that reported the divergence |
| SERVICE | VARCHAR2(64) | | Service name of the session that reported the divergence |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MODULE | VARCHAR2(64) | | Module name of the session that reported the divergence |
| ACTION | VARCHAR2(64) | | Action name of the session that reported the divergence |

# DBA_WORKLOAD_REPLAY_SCHEDULES

DBA_WORKLOAD_REPLAY_SCHEDULES displays the names of replay schedules for the current replay directory. A replay schedule defines one or multiple workload captures, and the order to start their replays. The current replay directory is set by DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY('replay_dir'). Each row in the view contains information about one replay schedule.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEDULE_NAME | VARCHAR2(128) | NOT NULL | The name of a schedule to be replayed. It defines one or multiple workload captures, and the order to start their replays. |
| DIRECTORY | VARCHAR2(128) | NOT NULL | Directory object name for the replay schedule name |
| STATUS | VARCHAR2(128) | | NEW if the schedule is being created, CURRENT if the schedule is currently being used by a replay, otherwise NULL |

**See Also:**

- "DBA_WORKLOAD_SCHEDULE_CAPTURES" displays the workload captures in a replay schedule.
- "DBA_WORKLOAD_SCHEDULE_ORDERING" displays the order to start captures in a replay schedule.
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_WORKLOAD_REPLAY package

# DBA_WORKLOAD_REPLAYS

DBA_WORKLOAD_REPLAYS displays all the workload replays that have been performed in the current database. It also lists replays on which DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO() has been called. Each row contains information about one workload replay.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ID | NUMBER | NOT NULL | Internal key for the workload replay |
| NAME | VARCHAR2(128) | NOT NULL | Name of the workload replay |
| DBID | NUMBER | NOT NULL | ID of the database in which the workload was replayed |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DBNAME | VARCHAR2(10) | NOT NULL | Name of the database in which the workload was replayed |
| DBVERSION | VARCHAR2(128) | NOT NULL | Version of the database in which the workload was replayed |
| PARALLEL | VARCHAR2(3) | | Indicates whether the database in which the workload was replayed was an Oracle RAC database (YES) or a single instance database (NO) |
| DIRECTORY | VARCHAR2(128) | NOT NULL | Name of the directory object for the workload replay |
| CAPTURE_ID | NUMBER | | ID of the capture (DBA_WORKLOAD_CAPTURES.ID) that was replayed. If the replay involves a replay schedule, the CAPTURE_ID will be null. |
| STATUS | VARCHAR2(40) | NOT NULL | Current status of the workload replay:<br>• PREPARE - Workload prepare has been started and is waiting for clients to join<br>• IN PROGRESS - Workload replay is in progress<br>• COMPLETED - Workload replay has successfully completed<br>• CANCELLED - Workload replay or the workload prepare has been cancelled<br>• FAILED - Workload replay was aborted due to errors encountered. See the COMMENTS column for further information. |
| PREPARE_TIME | DATE | | Datetime at which the workload prepare started |
| START_TIME | DATE | | Datetime when the replay began |
| END_TIME | DATE | | Datetime when the replay completed or cancelled; NULL if the replay is still in progress |
| DURATION_SECS | NUMBER | | Duration of the workload replay (in seconds) |
| NUM_CLIENTS | NUMBER | NOT NULL | Number of workload replay client processes that were used in this workload replay |
| NUM_CLIENTS_DONE | NUMBER | NOT NULL | Number of workload replay client processes that have finished replay |
| FILTER_SET_NAME | VARCHAR2(1000) | | Name of the filter set used for the replay |
| DEFAULT_ACTION | VARCHAR2(30) | NOT NULL | Reserved for future use |
| SYNCHRONIZATION | VARCHAR2(9) | | Indicates whether recorded transaction semantics should be maintained (TRUE) or not (FALSE)<br>When synchronization is on, the commit order observed during the original workload capture will be preserved. Every action that is replayed will be executed only after all of its dependent commits have been executed. Dependent |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | commits are commits that were issued before the given action in the original workload capture.<br>**See Also:** DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY( ) in *Oracle Database PL/SQL Packages and Types Reference* for a detailed explanation of this replay parameter |
| CONNECT_TIME_SCALE | NUMBER | NOT NULL | Connection time scaling factor for captured streams during replay. The value is interpreted as a percentage.The default value of 100 means 100 percent.<br>**See Also:** DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY( ) in *Oracle Database PL/SQL Packages and Types Reference* for a detailed explanation of this replay parameter |
| THINK_TIME_SCALE | NUMBER | NOT NULL | Think time scaling factor for captured streams during replay. It scales the thinking time elapsed between two successive user calls from the same captured stream. The input is interpreted as a percentage. The default value of 100 means 100 percent.<br>**See Also:** DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY( ) in *Oracle Database PL/SQL Packages and Types Reference* for a detailed explanation of this replay parameter |
| THINK_TIME_AUTO_CORRECT | VARCHAR2(5) | | Indicates whether the think time should be automatically corrected between calls (TRUE) or not (FALSE)<br>A value of TRUE reduces think time if replay goes slower than capture.<br>A value of FALSE results in no action.<br>**See Also:** DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY( ) in *Oracle Database PL/SQL Packages and Types Reference* for a detailed explanation of this replay parameter |
| SCALE_UP_MULTIPLIER | NUMBER | NOT NULL | Before the multiple-capture replay, SCALE_UP_MULTIPLIER is used to scale up the query part of a workload capture. The queries from each captured session are replayed concurrently as many times as the value of SCALE_UP_MULTIPLIER. |
| USER_CALLS | NUMBER | | Total number of user calls replayed |
| DBTIME | NUMBER | | Accumulated database time (in microseconds) for the replay |
| NETWORK_TIME | NUMBER | | Accumulated network time for the replay (in microseconds) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| THINK_TIME | NUMBER | | Accumulated think time (in microseconds) for the replay |
| PAUSE_TIME | NUMBER | | The total time (in seconds) that the replay has been paused (by calling the PAUSE_REPLAY procedure) |
| ELAPSED_TIME_DIFF | NUMBER | | Reserved for future use |
| AWR_DBID | NUMBER | | Database ID of the AWR snapshots that correspond to this workload replay. For replays that were performed in the current database, this value is equal to the current database's DBID. For replays that were performed in other databases, this value will either be NULL or will be populated by `DBMS_WORKLOAD_REPLAY.IMPORT_AWR()`.<br>**See Also:** DBMS_WORKLOAD_REPLAY.IMPORT_AWR() in *Oracle Database PL/SQL Packages and Types Reference* |
| AWR_BEGIN_SNAP | NUMBER | | Begin snapshot ID of the AWR snapshots that correspond to this workload replay |
| AWR_END_SNAP | NUMBER | | End snapshot ID of the AWR snapshots that correspond to this workload replay |
| AWR_EXPORTED | VARCHAR2(12) | | Indicates whether the AWR snapshots that correspond to this workload replay have been exported using `DBMS_WORKLOAD_REPLAY.EXPORT_AWR()` (`YES`) or not (`NO`), or whether AWR snapshots cannot be exported because the replay is still in progress, has run to completion successfully, or was done in a different database from which it was not exported (`NOT POSSIBLE`)<br>**See Also:** DBMS_WORKLOAD_REPLAY.EXPORT_AWR() in *Oracle Database PL/SQL Packages and Types Reference* |
| ERROR_CODE | NUMBER | | Error code for this workload replay |
| ERROR_MESSAGE | VARCHAR2(300) | | Error message for this workload replay |
| DIR_PATH | VARCHAR2(4000) | NOT NULL | Full directory path for the replay directory object |
| REPLAY_DIR_NUMBER | NUMBER | | A hash value computed based on the values of other columns in this view, such as the `NAME`, `DBID`, `DBNAME`, `PREPARE_TIME`, `START_TIME`, and `END_TIME` columns. It should be fairly unique for any replay. The value is used to create a subdirectory under the replay directory. |
| SQLSET_OWNER | VARCHAR2(128) | | User name of the SQL tuning set owner |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SQLSET_NAME | VARCHAR2(128) | | Name of the SQL tuning set for this workload replay |
| SCHEDULE_NAME | VARCHAR2(128) | | The name of a schedule to be replayed. It defines one or multiple workload captures, and the order to start their replays.<br>If SCHEDULE_NAME is NULL, the replay is a regular replay introduced since Oracle Database 11*g*, done with existing APIs from DBMS_WORKLOAD_REPLAY: INITIALIZE_REPLAY, PREPARE_REPLAY, and START_REPLAY.<br>If SCHEDULE_NAME is not NULL, the replay is a new consolidated replay introduced in Oracle Database 12*c*. That is, it is a replay of one or more workload captures done with new APIs at DBMS_WORKLOAD_REPLAY: INITIALIZE_CONSOLIDATED_REPLAY, PREPARE_CONSOLIDATED_REPLAY, and START_CONSOLIDATED_REPLAY.<br>**See Also:** DBMS_WORKLOAD_REPLAY in *Oracle Database PL/SQL Packages and Types Reference* |
| DIVERGENCE_LOAD_STATUS | VARCHAR2(5) | | Indicates whether replay divergence data have been loaded (TRUE) or not (FALSE) |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_WORKLOAD_REPLAY package

# DBA_WORKLOAD_SCHEDULE_CAPTURES

DBA_WORKLOAD_SCHEDULE_CAPTURES displays the workload captures used by replay schedules. Each row in the view contains information about one workload capture.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEDULE_NAME | VARCHAR2(128) | NOT NULL | The name of a schedule to be replayed |
| SCHEDULE_CAP_ID | NUMBER | NOT NULL | Identifies a workload capture added to a replay schedule. It starts with 1. If the same capture is added multiple times to a schedule, there will be multiple rows with different SCHEDULE_CAP_ID columns and identical CAPTURE_ID columns. |
| CAPTURE_ID | NUMBER | NOT NULL | Points to the capture ID from DBA_WORKLOAD_CAPTURES. |
| CAPTURE_DIR | VARCHAR2(128) | NOT NULL | Name of the directory object for workload capture |
| OS_SUBDIR | VARCHAR2(4000) | NOT NULL | Name of the subdirectory under the replay directory for this workload capture |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| MAX_CONCURRENT_SESSIONS | NUMBER | | The maximal number of concurrent sessions that was seen in this workload capture |
| NUM_CLIENTS_ASSIGNED | NUMBER | | Number of clients assigned to this workload capture before replay starts |
| NUM_CLIENTS | NUMBER | | Number of clients that are running for this workload capture during replay |
| NUM_CLIENTS_DONE | NUMBER | | Number of clients that have finished the replay of this workload capture |
| STOP_REPLAY | VARCHAR2(1) | NOT NULL | Indicates whether the whole replay will stop once the replay of this workload capture is done (Y) or not (N) |
| TAKE_BEGIN_SNAPSHOT | VARCHAR2(1) | NOT NULL | Indicates whether an AWR snapshot will be taken when the replay of this capture starts (Y) or not (N) |
| TAKE_END_SNAPSHOT | VARCHAR2(1) | NOT NULL | Indicates whether an AWR snapshot will be taken when the replay of this capture finishes (Y) or not (N) |
| QUERY_ONLY | VARCHAR2(1) | NOT NULL | Indicates whether only the query-only workload from the current workload capture will be replayed, skipping all the DML/DDL that might update the database (Y) or not (N) |
| START_DELAY_SECS | NUMBER | | Displays the wait time (in seconds) when the replay of a workload capture is ready to start. "Ready to start" means the capture does not wait for any other capture, or all the captures for which it should wait have already been replayed. The default value is 0. |
| START_TIME | DATE | | Start time for the replay of this capture |
| END_TIME | DATE | | Finish time for the replay of this capture |
| AWR_DBID | NUMBER | | AWR database ID of the replay |
| AWR_BEGIN_SNAP | NUMBER | | AWR snapshot ID when the replay starts |
| AWR_END_SNAP | NUMBER | | AWR snapshot ID when the replay finishes |

# DBA_WORKLOAD_SCHEDULE_ORDERING

DBA_WORKLOAD_SCHEDULE_ORDERING displays the start ordering between workload captures in the replay schedule. Each row in the view defines one start ordering between two workload captures in the same replay schedule.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SCHEDULE_NAME | VARCHAR2(128) | NOT NULL | Name of a schedule to be replayed |
| SCHEDULE_CAP_ID | NUMBER | NOT NULL | Identifies the workload capture that will wait |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| WAITFOR_CAP_ID | NUMBER | NOT NULL | Identifies the workload capture for which the workload capture identified by SCHEDULE_CAP_ID needs to wait. The replay of capture SCHEDULE_CAP_ID will not start until capture WAITFOR_CAP_ID finishes its replay.<br>If the view has multiple rows with the same SCHEDULE_CAP_ID but different WAITFOR_CAP_ID, it defines a schedule so that the replay of a capture specified by SCHEDULE_CAP_ID will not start unless all the replays of the waited captures run into completion.<br>If the view has multiple rows with the same WAITFOR_CAP_ID but different SCHEDULE_CAP_ID, it defines a schedule so that the replay of multiple captures will not start unless the replay of the capture specified by WAITFOR_CAP_ID finishes. |

# DBA_WORKLOAD_USER_MAP

DBA_WORKLOAD_USER_MAP contains all the mappings ever done until they are removed at some point.

The mappings are stored in a table made public through this view.

To remove old mappings, execute this statement:

```
SQL> delete * from DBA_WORKLOAD_USER_MAP;
```

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REPLAY_ID | NUMBER | | This is a foreign key to the ID column in the DBA_WORKLOAD_REPLAYS view |
| SCHEDULE_CAP_ID | NUMBER | | The ID of a capture in the schedule |
| CAPTURE_USER | VARCHAR2(4000) | NOT NULL | The user name during the time of the workload capture |
| REPLAY_USER | VARCHAR2(4000) | | The user name to which the captured user should be remapped during replay. If the REPLAY_USER is null, the CAPTURE_USER is used during replay. In other words, the original user is used. |

**See Also:**

"DBA_WORKLOAD_REPLAYS"

# DBA_XML_INDEXES

DBA_XML_INDEXES describes all XML indexes in the database. Its columns are the same as those in ALL_XML_INDEXES.

**See Also:**

"ALL_XML_INDEXES"

## DBA_XML_NESTED_TABLES

DBA_XML_NESTED_TABLES describes all the tables and their corresponding nested tables. Its columns are the same as those in ALL_XML_NESTED_TABLES.

**See Also:**

"ALL_XML_NESTED_TABLES"

## DBA_XML_OUT_OF_LINE_TABLES

DBA_XML_OUT_OF_LINE_TABLES descibes all the out of line tables connected to a given root table for the same schema. Its columns are the same as those in ALL_XML_OUT_OF_LINE_TABLES.

**See Also:**

"ALL_XML_OUT_OF_LINE_TABLES"

## DBA_XML_SCHEMA_ATTRIBUTES

DBA_XML_SCHEMA_ATTRIBUTES describes all the attributes and their properties. Its columns are the same as those in ALL_XML_SCHEMA_ATTRIBUTES.

**See Also:**

"ALL_XML_SCHEMA_ATTRIBUTES"

## DBA_XML_SCHEMA_COMPLEX_TYPES

DBA_XML_SCHEMA_COMPLEX_TYPES describes all complex types in the database. Its columns are the same as those in ALL_XML_SCHEMA_COMPLEX_TYPES.

**See Also:**

"ALL_XML_SCHEMA_COMPLEX_TYPES"

## DBA_XML_SCHEMA_ELEMENTS

DBA_XML_SCHEMA_ELEMENTS describes all the elements and their properties. Its columns are the same as those in ALL_XML_SCHEMA_ELEMENTS.

**See Also:**

"ALL_XML_SCHEMA_ELEMENTS"

## DBA_XML_SCHEMA_NAMESPACES

DBA_XML_SCHEMA_NAMESPACES describes all the available namespaces. Its columns are the same as those in ALL_XML_SCHEMA_NAMESPACES.

**See Also:**

"ALL_XML_SCHEMA_NAMESPACES"

## DBA_XML_SCHEMA_SIMPLE_TYPES

DBA_XML_SCHEMA_SIMPLE_TYPES describes all simple types. Its columns are the same as those in ALL_XML_SCHEMA_SIMPLE_TYPES.

**See Also:**

"ALL_XML_SCHEMA_SIMPLE_TYPES"

## DBA_XML_SCHEMA_SUBSTGRP_HEAD

DBA_XML_SCHEMA_SUBSTGRP_HEAD describes the heads of substitution groups. Its columns are the same as those in ALL_XML_SCHEMA_SUBSTGRP_HEAD.

**See Also:**

"ALL_XML_SCHEMA_SUBSTGRP_HEAD"

## DBA_XML_SCHEMA_SUBSTGRP_MBRS

DBA_XML_SCHEMA_SUBSTGRP_MBRS describes all members of substitution groups. Its columns are the same as those in ALL_XML_SCHEMA_SUBSTGRP_MBRS.

**See Also:**

"ALL_XML_SCHEMA_SUBSTGRP_MBRS"

## DBA_XML_SCHEMAS

DBA_XML_SCHEMAS describes all registered XML schemas in the database. Its columns are the same as those in ALL_XML_SCHEMAS.

**See Also:**

"ALL_XML_SCHEMAS"

## DBA_XML_TAB_COLS

DBA_XML_TAB_COLS describes the columns of all XML tables in the database. Its columns are the same as those in ALL_XML_TAB_COLS.

**See Also:**

"ALL_XML_TAB_COLS"

## DBA_XML_TABLES

DBA_XML_TABLES describes all XML tables in the database. Its columns are the same as those in ALL_XML_TABLES.

**See Also:**

"ALL_XML_TABLES"

## DBA_XML_VIEW_COLS

DBA_XML_VIEW_COLS describes the columns of all XML views in the database. Its columns are the same as those in ALL_XML_VIEW_COLS.

**See Also:**

"ALL_XML_VIEW_COLS"

## DBA_XML_VIEWS

DBA_XML_VIEWS describes all XML views in the database. Its columns are the same as those in ALL_XML_VIEWS.

**See Also:**

"ALL_XML_VIEWS"

## DBA_XS_AUDIT_POLICY_OPTIONS

DBA_XS_AUDIT_POLICY_OPTIONS describes auditing options defined under all audit policies specific to Oracle Database Real Application Security.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| POLICY_NAME | VARCHAR2(128) | | Name of the audit policy |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AUDIT_CONDITION | VARCHAR2(4000) | | Condition associated with the audit policy |
| AUDIT_OPTION | VARCHAR2(128) | | Auditing option defined in the audit policy |
| CONDITION_EVAL_OPT | VARCHAR2(9) | | Evaluation option associated with the audit policy's condition. The possible values are STATEMENT, SESSION, INSTANCE, NONE. |
| COMMON | VARCHAR2(3) | | Indicates whether the audit policy is a common audit policy (YES) or local (NO). The value is NULL in non-CDBs. |

**See Also:**

*Oracle Database Security Guide* for more information about auditing

# DBA_XS_AUDIT_TRAIL

DBA_XS_AUDIT_TRAIL describes all audit records specific to Oracle Database Real Application Security.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERID | VARCHAR2(30) | | Name of the database user whose actions were audited |
| ACTION | NUMBER | | Numeric audit trail action type code. The corresponding name of the action type is in the ACTION_NAME column. |
| ACTION_NAME | VARCHAR2(64) | | Name of the action type corresponding to the numeric code in the ACTION column |
| OBJ_OWNER | VARCHAR2(30) | | Owner of the object affected by the action |
| OBJ_NAME | VARCHAR2(128) | | Name of the object affected by the action |
| RETURN_CODE | NUMBER | | Oracle error code generated by the action |
| XS_USER_NAME | VARCHAR2(128) | | Name of the Real Application Security user |
| XS_SESSIONID | RAW(33) | | Identifer of the Real Application Security session |
| XS_INACTIVITY_TIMEOUT | NUMBER | | Inactivity timeout of the Real Application Security session |
| XS_ENTITY_TYPE | VARCHAR2(32) | | Type of the Real Application Security entity. Possible values are USER, ROLE, ROLESET, SECURITYCLASS, ACL, DATASECURITY, and NSTEMPLATE. |
| XS_TARGET_PRINCIPAL_NAME | VARCHAR2(30) | | Target principal name in Real Application Security operations. Possible operations are set verifier, set password, add proxy, remove proxy, switch user, assign user, create session, grant roles. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| XS_PROXY_USER_NAME | VARCHAR2(30) | | Name of the Real Application Security proxy user. |
| XS_DATASEC_POLICY_NAME | VARCHAR2(30) | | Name of the Real Application Security data security policy enabled or disabled |
| XS_SCHEMA_NAME | VARCHAR2(30) | | Name of the schema in enable, disable data security policy and global callback operation |
| XS_CALLBACK_EVENT_TYPE | VARCHAR2(32) | | Real Application Security global callback event type |
| XS_PACKAGE_NAME | VARCHAR2(30) | | Real Application Security callback package name for the global callback |
| XS_PROCEDURE_NAME | VARCHAR2(30) | | Real Application Security callback procedure name for the global callback |
| XS_ENABLED_ROLE | VARCHAR2(30) | | The role that is enabled |
| XS_COOKIE | VARCHAR2(1024) | | Real Application Security session cookie |
| XS_NS_NAME | VARCHAR2(30) | | Name of the Real Application Security session namespace |
| XS_NS_ATTRIBUTE | VARCHAR2(4000) | | Name of the Real Application Security session namespace attribute |
| XS_NS_ATTRIBUTE_OLD_VAL | VARCHAR2(4000) | | The old value of the Real Application Security session namespace attribute |
| XS_NS_ATTRIBUTE_NEW_VAL | VARCHAR2(4000) | | The new value of the Real Application Security session namespace attribute |
| EVENT_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | | Timestamp of audit record |

**See Also:**

*Oracle Database Security Guide* for more information about auditing

# DBA_XS_ENB_AUDIT_POLICIES

DBA_XS_ENB_AUDIT_POLICIES describes all the audit policies specific to Oracle Database Real Application Security that are enabled to users.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USER_NAME | VARCHAR2(128) | | Database username for whom the audit policy is enabled. If the audit policy is enabled for all users, then the value is ALL USERS. |
| POLICY_NAME | VARCHAR2(128) | | Name of the audit policy |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ENABLED_OPT | VARCHAR2(8) | | Enabled option of the audit policy. The possible values are BY, EXCEPT, and DISABLED. |
| SUCCESS | VARCHAR2(3) | | Indicates whether the audit policy is enabled for auditing successful events (YES) or not (NO) |
| FAILURE | VARCHAR2(3) | | Indicates whether the audit policy is enabled for auditing unsuccessful events (YES) or not (NO) |

**See Also:**

*Oracle Database Security Guide* for more information about auditing

## DBA_XSTREAM_ADMINISTRATOR

DBA_XSTREAM_ADMINISTRATOR displays information about the users who have been granted privileges to be XStream administrators by procedures in the DBMS_XSTREAM_AUTH package. Its columns are the same as those in ALL_XSTREAM_ADMINISTRATOR.

**See Also:**

"ALL_XSTREAM_ADMINISTRATOR"

## DBA_XSTREAM_INBOUND

DBA_XSTREAM_INBOUND displays information about all XStream inbound servers in the database. Its columns are the same as those in ALL_XSTREAM_INBOUND.

**See Also:**

"ALL_XSTREAM_INBOUND"

## DBA_XSTREAM_INBOUND_PROGRESS

DBA_XSTREAM_INBOUND_PROGRESS displays information about the progress made by all XStream inbound servers in the database. Its columns are the same as those in ALL_XSTREAM_INBOUND_PROGRESS.

**See Also:**

"ALL_XSTREAM_INBOUND_PROGRESS"

## DBA_XSTREAM_OUT_SUPPORT_MODE

DBA_XSTREAM_OUT_SUPPORT_MODE displays information about the level of XStream capture process support for the tables in the database. Its columns are the same as those in ALL_XSTREAM_OUT_SUPPORT_MODE.

**See Also:**

"ALL_XSTREAM_OUT_SUPPORT_MODE"

## DBA_XSTREAM_OUTBOUND

DBA_XSTREAM_OUTBOUND displays information about all XStream outbound servers in the database. Its columns are the same as those in ALL_XSTREAM_OUTBOUND.

**See Also:**

"ALL_XSTREAM_OUTBOUND"

## DBA_XSTREAM_OUTBOUND_PROGRESS

DBA_XSTREAM_OUTBOUND_PROGRESS displays information about the progress made by all XStream outbound servers in the database. Its columns are the same as those in ALL_XSTREAM_OUTBOUND_PROGRESS.

**See Also:**

"ALL_XSTREAM_OUTBOUND_PROGRESS"

## DBA_XSTREAM_RULES

DBA_XSTREAM_RULES displays information about all XStream rules in the database. Its columns are the same as those in ALL_XSTREAM_RULES.

**See Also:**

"ALL_XSTREAM_RULES"

## DBA_XSTREAM_SPLIT_MERGE

DBA_XSTREAM_SPLIT_MERGE displays information about XStream current automatic split and merge operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ORIGINAL_CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the original capture process |
| CLONED_CAPTURE_NAME | VARCHAR2(128) | | Name of the cloned capture process |
| ORIGINAL_CAPTURE_STATUS | VARCHAR2(8) | | Status of the original capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLONED_CAPTURE_STATUS | VARCHAR2(8) | | Status of the cloned capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| ORIGINAL_XSTREAM_NAME | VARCHAR2(128) | | Name of the original XStream component that receives database changes directly from the original capture process. The component is either a progagation or a local apply process. |
| CLONED_XSTREAM_NAME | VARCHAR2(128) | | Name of the cloned XStream component that receives database changes directly from the cloned capture process. The component is either a progagation or a local apply process. |
| XSTREAM_TYPE | VARCHAR2(11) | | Type of the component in ORIGINAL_XSTREAM_NAME and CLONED_XSTREAM_NAME:<br>• PROPAGATION<br>• APPLY |
| RECOVERABLE_SCRIPT_ID | RAW(16) | | Unique ID of the script to split or merge operation |
| SCRIPT_STATUS | VARCHAR2(12) | | Status of the recoverable script:<br>• GENERATING<br>• NOT EXECUTING<br>• EXECUTING<br>• EXECUTED<br>• ERROR |
| ACTION_TYPE | VARCHAR2(7) | | type of action performed by the script:<br>• SPLIT<br>• MERGE<br>• MONITOR |
| ACTION_THRESHOLD | VARCHAR2(40) | | For SPLIT actions, the threshold set by the split_threshold capture process parameter. For MERGE actions, the threshold set by the merge_threshold capture process parameter. |
| STATUS | VARCHAR2(16) | | Status of the action:<br>• NOTHING TO SPLIT - Not ready to split or does not need to split<br>• ABOUT TO SPLIT<br>• SPLITTING - A split is in progress<br>• SPLIT DONE - A split is done<br>• NOTHING TO MERGE - Not ready to merge<br>• ABOUT TO MERGE<br>• MERGING - A merge is in progress<br>• MERGE DONE - A merge is done<br>• ERROR - An error was returned during a split or merge<br>• NONSPLITTABLE - The original capture is not splittable either because it is disabled, it has |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | more than one publisher to its queue, or it has only one destination for captured messages |
| STATUS_UPDATE_TIME | TIMESTAMP(6) | | Time when status was last updated |
| CREATION_TIME | TIMESTAMP(6) | | Time when the action started |
| LAG | NUMBER | | Time (in seconds) that the cloned capture process lags behind the original capture process |
| JOB_OWNER | VARCHAR2(128) | | Owner of the job that performs the split or merge operation |
| JOB_NAME | VARCHAR2(128) | | Name of the job that performs the split or merge operation |
| JOB_STATE | VARCHAR2(15) | | Current state of the job:<br>• DISABLED<br>• RETRY SCHEDULED<br>• SCHEDULED<br>• RUNNING<br>• COMPLETED<br>• BROKEN<br>• FAILED<br>• REMOTE<br>• SUCCEEDED<br>• CHAIN_STALLED |
| JOB_NEXT_RUN_DATE | VARCHAR2(64) | | Next time the job will run |
| ERROR_NUMBER | NUMBER | | Error number if the capture process was aborted |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message if the capture process was aborted |

# DBA_XSTREAM_SPLIT_MERGE_HIST

DBA_XSTREAM_SPLIT_MERGE_HIST displays information about past XStream automatic split and merge operations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ORIGINAL_CAPTURE_NAME | VARCHAR2(128) | NOT NULL | Name of the original capture process |
| CLONED_CAPTURE_NAME | VARCHAR2(128) | | Name of the cloned capture process |
| ORIGINAL_QUEUE_OWNER | VARCHAR2(128) | | Owner of the queue used by the original capture process |
| ORIGINAL_QUEUE_NAME | VARCHAR2(128) | | Name of the queue used by the original capture process |
| CLONED_QUEUE_OWNER | VARCHAR2(128) | | Owner of the queue used by the cloned capture process |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CLONED_QUEUE_NAME | VARCHAR2(128) | | Name of the queue used by the cloned capture process |
| ORIGINAL_CAPTURE_STATUS | VARCHAR2(8) | | Status of the original capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| CLONED_CAPTURE_STATUS | VARCHAR2(8) | | Status of the cloned capture process:<br>• DISABLED<br>• ENABLED<br>• ABORTED |
| ORIGINAL_XSTREAM_NAME | VARCHAR2(128) | | Name of the original XStream component that receives database changes directly from the original capture process. The component is either a progagation or a local apply process. |
| CLONED_XSTREAM_NAME | VARCHAR2(128) | | Name of the cloned XStream component that receives database changes directly from the cloned capture process. The component is either a progagation or a local apply process. |
| XSTREAM_TYPE | VARCHAR2(11) | | Type of the component in ORIGINAL_XSTREAM_NAME and CLONED_XSTREAM_NAME:<br>• PROPAGATION<br>• APPLY |
| RECOVERABLE_SCRIPT_ID | RAW(16) | | Unique ID of the script to split or merge operation |
| SCRIPT_STATUS | VARCHAR2(12) | | Status of the recoverable script:<br>• GENERATING<br>• NOT EXECUTING<br>• EXECUTING<br>• EXECUTED<br>• ERROR |
| ACTION_TYPE | VARCHAR2(7) | | type of action performed by the script:<br>• SPLIT<br>• MERGE<br>• MONITOR |
| ACTION_THRESHOLD | VARCHAR2(40) | | For SPLIT actions, the threshold set by the split_threshold capture process parameter. For MERGE actions, the threshold set by the merge_threshold capture process parameter. |
| STATUS | VARCHAR2(16) | | Status of the action:<br>• NOTHING TO SPLIT - Not ready to split or does not need to split<br>• ABOUT TO SPLIT<br>• SPLITTING - A split is in progress<br>• SPLIT DONE - A split is done<br>• NOTHING TO MERGE - Not ready to merge |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • ABOUT TO MERGE<br>• MERGING - A merge is in progress<br>• MERGE DONE - A merge is done<br>• ERROR - An error was returned during a split or merge<br>• NONSPLITTABLE - The original capture is not splittable either because it is disabled, it has more than one publisher to its queue, or it has only one destination for captured messages |
| STATUS_UPDATE_TIME | TIMESTAMP(6) | | Time when status was last updated |
| CREATION_TIME | TIMESTAMP(6) | | Time when the action started |
| LAG | NUMBER | | Time (in seconds) that the cloned capture process lags behind the original capture process |
| JOB_OWNER | VARCHAR2(128) | | Owner of the job that performs the split or merge operation |
| JOB_NAME | VARCHAR2(128) | | Name of the job that performs the split or merge operation |
| ERROR_NUMBER | NUMBER | | Error number if the capture process was aborted |
| ERROR_MESSAGE | VARCHAR2(4000) | | Error message if the capture process was aborted |

# DBA_XSTREAM_STMT_HANDLERS

DBA_XSTREAM_STMT_HANDLERS displays information about all XStream statement DML handlers in the database..

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HANDLER_NAME | VARCHAR2(128) | NOT NULL | Name of the statement handler |
| HANDLER_COMMENT | VARCHAR2(4000) | | Comment of the statement handler |
| CREATION_TIME | TIMESTAMP(6) | | Timestamp for script creation |
| MODIFICATION_TIME | TIMESTAMP(6) | | Timestamp for script modification |

# DBA_XSTREAM_STMTS

DBA_XSTREAM_STMTS displays information about the statements in all XStream statement DML handlers in the database..

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HANDLER_NAME | VARCHAR2(128) | NOT NULL | Name of the statement handler |
| EXECUTION_SEQUENCE | NUMBER | NOT NULL | Execution sequence of the statement |
| STATEMENT | CLOB | | Text of the SQL statement |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CREATION_TIME | TIMESTAMP(6) | | Timestamp for statement creation |
| MODIFICATION_TIME | TIMESTAMP(6) | | Timestamp for statement modification |

# DBA_XSTREAM_TRANSFORMATIONS

DBA_XSTREAM_TRANSFORMATIONS displays information about all XStream transformations available on a system, in order of execution. Its columns are the same as those in ALL_XSTREAM_TRANSFORMATIONS.

**See Also:**

"ALL_XSTREAM_TRANSFORMATIONS"

# DBA_ZONEMAP_MEASURES

DBA_ZONEMAP_MEASURES describes the measures for all the zone maps in the database. Its columns are the same as those in ALL_ZONEMAP_MEASURES.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2), and is intended for use with Oracle Exadata release 12.1.2.1.1 or later.

**See Also:**

- "ALL_ZONEMAP_MEASURES"
- *Oracle Database Data Warehousing Guide* for more information about zone maps

# DBA_ZONEMAPS

DBA_ZONEMAPS describes all the zone maps in the database. Its columns are the same as those in ALL_ZONEMAPS.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2), and is intended for use with Oracle Exadata release 12.1.2.1.1 or later.

**See Also:**

- "ALL_ZONEMAPS"
- *Oracle Database Data Warehousing Guide* for more information about zone maps

# DBFS_CONTENT

DBFS_CONTENT displays all the path items from all available content stores in the system.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STORE | VARCHAR2(256) | | Name of store |
| MOUNT | VARCHAR2(256) | | Location at which instance of store is mounted |
| PATHNAME | VARCHAR2(1024) | | Name of path to item |
| PATHTYPE | VARCHAR2(32) | | Type of path item (see DBMS_DBFS_CONTENT Constants - Path Name Types) |
| FILEDATA | BLOB | | BLOB locator that can be used to access data in the path item |
| STD_ACCESS_TIME | TIMESTAMP(6) | | Time of last access of a path name's contents |
| STD_ACL | VARCHAR2(1024) | | Access Control List (in standard ACL syntax) |
| STD_CHANGE_TIME | TIMESTAMP(6) | | Time of last change to the path name |
| STD_CHILDREN | NUMBER | | Number of child directories/folders a directory/folder path has (this property should be available in providers that support the FEATURE_FOLDERS feature) |
| STD_CONTENT_TYPE | VARCHAR2(1024) | | One or more client-supplied mime-types (in standard RFC syntax) describing the path name which is typically of TYPE_FILE. The content type is not necessarily interpreted by the store. |
| STD_CREATION_TIME | TIMESTAMP(6) | | Time at which the item was created. Once set, this value remains the same for the lifetime of the path name. |
| STD_DELETED | NUMBER | | Set to a nonzero number if the path name has been soft-deleted but not yet purged (see DBMS_DBFS_CONTENT Constants - Store Features) |
| STD_GUID | NUMBER | | Store-specific unique identifier for a path name. Clients must not depend on the GUID being unique across different stores, but a given (store-name, store-specific-path name) has a stable and unique GUID for its lifetime. |
| STD_MODIFICATION_TIME | TIMESTAMP(6) | | Time of last change to the data associated with a path name. Changes to the content of a TYPE_FILE or TYPE_REFERENCE path, the referent of the TYPE_LINK path, and addition or deletion of immediate children in a TYPE_DIRECTORY path, all constitute data changes. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STD_OWNER | VARCHAR2(32) | | Client-supplied (or implicit) owner name for the path name |
| STD_PARENT_GUID | NUMBER | | Store-specific unique identifier for the parent of a path name. Clients must not depend on the GUID being unique across different stores, but a given (store-name, store-specific-path name) has a stable and unique GUID for its lifetime.<br>The GUID of the parent of this path name (that is that std_parent_guid(pathname) == std_guid(parent(pathname))). |
| STD_REFERENT | VARCHAR2(1024) | | Content of the symbolic link of a TYPE_LINK path, otherwise NULL. The STD_REFERENT can be an arbitrary string and must not necessarily be interpreted as path name by clients (or such interpretation should be done with great care). |
| OPT_HASH_TYPE | VARCHAR2(32) | | Type of hash provided in the OPT_HASH_VALUE property (see DBMS_CRYPTO for possible options) |
| OPT_HASH_VALUE | VARCHAR2(128) | | Hash value of type OPT_HASH_TYPE describing the content of the path name |
| OPT_LOCK_COUNT | NUMBER | | Number of compatible locks placed on a path name. If different principals are allowed to place compatible (read) locks on a path, the OPT_LOCKER must specify all lockers with repeats so that lock counts can be correctly maintained. |
| OPT_LOCK_DATA | VARCHAR2(128) | | Client-supplied user-data associated with a user-lock, uninterpreted by the store |
| OPT_LOCKER | VARCHAR2(128) | | One or more implicit or client-specified principals that applied a user-lock on a path name |
| OPT_LOCK_STATUS | NUMBER | | One of the (LOCK_READ_ONLY, LOCK_WRITE_ONLY, LOCK_READ_WRITE) values describing the type of lock currently applied on a path name |
| OPT_VERSION | NUMBER | | Sequence number for linear versioning of a path name |
| OPT_VERSION_PATH | VARCHAR2(1024) | | Version path name for hierarchical versioning of a path name |
| OPT_CONTENT_ID | RAW(128) | | A provider-generated store-specific unique contentID in the form of a string for a file element (that may optionally not be associated with a path (see FEATURE_CONTENT_ID and FEATURE_LAZY_PATH in DBMS_DBFS_CONTENT Constants - Store Features) |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about DBMS_DBFS_CONTENT Constants - Path Name Types and DBMS_DBFS_CONTENT Constants - Store Features
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_CRYPTO package

# DBFS_CONTENT_PROPERTIES

DBFS_CONTENT_PROPERTIES displays all the property/value pairs for all path items in all content stores in the system.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STORE | VARCHAR2(256) | | Name of store |
| MOUNT | VARCHAR2(256) | | Location at which instance of store is mounted |
| PATHNAME | VARCHAR2(1024) | | Name of path to item |
| PROPERTY_NAME | VARCHAR2(32) | | Name of the property |
| PROPERTY_VALUE | VARCHAR2(1024) | | Value of the property |
| PROPERTY_TYPE | NUMBER | | PL/SQL typecode for the property value |

# DBMS_ALERT_INFO

DBMS_ALERT_INFO describes registered alerts.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(30) | NOT NULL | Name of the alert |
| SID | VARCHAR2(30) | NOT NULL | Session ID of a session waiting for this alert |
| CHANGED | VARCHAR2(1) | | Boolean flag to indicate that an alert has been signaled. Y: alert signaled, N: no alert. |
| MESSAGE | VARCHAR2(1800) | | Optional message passed by signaler |

# DBMS_LOCK_ALLOCATED

DBMS_LOCK_ALLOCATED describes user-allocated locks.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | NOT NULL | Name of the lock |
| LOCKID | NUMBER(38) | | Lock identifier number |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EXPIRATION | DATE | | Planned lock expiration date (updates whenever the allocation procedure is run) |

# DEPTREE

This view, created by `utldtree.sql`, contains information on the object dependency tree. For user SYS, this view displays shared cursors (and only shared cursors) that depend on the object. For all other users, it displays objects other than shared cursors. Other users can access SYS.DEPTREE for information on shared cursors.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NESTED_LEVEL | NUMBER | | Nesting level in the dependency tree |
| TYPE | VARCHAR2(23) | | Object type |
| SCHEMA | VARCHAR2(128) | | Object schema |
| NAME | VARCHAR2(1002) | | Object name |
| SEQ# | NUMBER | | Sequence number in the dependency tree. Used for ordering queries.<br>**See Also:** "IDEPTREE" |

# DICT

DICT is a synonym for DICTIONARY.

**See Also:**

"DICTIONARY"

# DICT_COLUMNS

DICT_COLUMNS contains descriptions of columns in data dictionary tables and views.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | | Name of the object that contains the column |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column |
| COMMENTS | VARCHAR2(4000) | | Text comment on the column |

# DICTIONARY

DICTIONARY contains descriptions of data dictionary tables and views.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2(128) | | Name of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMMENTS | VARCHAR2(4000) | | Text comment on the object |

# DM_USER_MODELS

DM_USER_MODELS displays information about the models in the user's schema.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAME | VARCHAR2(128) | NOT NULL | Name of the model |
| FUNCTION_NAME | VARCHAR2(30) | | Model function:<br>• association - Association is a descriptive mining function. An association model identifies relationships and the probability of their occurrence within a data set.<br>• attribute_importance - Attribute Importance is a predictive mining function. An attribute importance model identifies the relative importance of an attribute in predicting a given outcome.<br>• classification - Classification is a predictive mining function. A classification model uses historical data to predict new discrete or categorical data.<br>The classification function can also be used for anomaly detection. In this case, the SVM algorithm with a null target is used (One-Class SVM).<br>• clustering - Clustering is a descriptive mining function. A clustering model identifies natural groupings within a data set.<br>• feature_extraction - Feature Extraction is a descriptive mining function. A feature extraction model creates an optimized data set on which to base a model.<br>• regression - Regression is a predictive mining function. A regression model uses historical data to predict new continuous, numeric data. |
| ALGORITHM_NAME | VARCHAR2(30) | | Algorithm used by the model:<br>• algo_name - Setting that specifies the algorithm used by the model.<br>• asso_max_rule_length - Setting that specifies the maximum length of a rule used by an association model.<br>• asso_min_confidence - Setting that specifies the minimum confidence for an association model.<br>• asso_min_support - Setting that specifies the minimum support for an association model.<br>• clas_cost_table_name - Setting that specifies the name of the cost matrix table for a classification model. |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • clas_priors_table_name - Setting that specifies the name of the prior probability table for NB and ABN models. Decision Tree is the only classification algorithm that does not use priors.<br>For SVM classification models, this setting specifies the name of a table of weights.<br>• clus_num_clusters - Setting that specifies the number of clusters for a clustering model.<br>• feat_num_features - Setting that specifies the number of features for a feature selection model. |
| CREATION_DATE | DATE | NOT NULL | Date on which the model was created |
| BUILD_DURATION | NUMBER | | Duration of the model build process |
| TARGET_ATTRIBUTE | VARCHAR2(128) | | Attribute designated as the target of a classification model |
| MODEL_SIZE | NUMBER | | Size of the model (in megabytes) |

# DOCUMENT_LINKS

DOCUMENT_LINKS provides system information about Oracle XML DB document links in Oracle XML DB Repository documents. When an XML document that includes XLink or XInclude links is added to the repository, these links can be mapped to document links, which are tracked using view DOCUMENT_LINKS.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SOURCE_ID | RAW(16) | | The source resource OID |
| TARGET_ID | RAW(16) | | The target resource OID |
| TARGET_PATH | VARCHAR2(4000) | | This column is always NULL. It is reserved for future use. |
| LINK_TYPE | VARCHAR2(8) | | The document link type: Hard or Weak |
| LINK_FORM | VARCHAR2(8) | | Whether the original link was of form XLink or XInclude |
| SOURCE_TYPE | VARCHAR2(17) | | Whether the link is contained in Resource Content or Resource Metadata |

**See Also:**

*Oracle XML DB Developer's Guide* for information about using this view

# ERROR_SIZE

ERROR_SIZE is accessed to create "DBA_OBJECT_SIZE" and "USER_OBJECT_SIZE".

# EXCEPTIONS

EXCEPTIONS contains information on violations of integrity constraints. This table is created by the utlexcpt.sql script.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROW_ID | ROWID | | Row that caused the violation |
| OWNER | VARCHAR2(128) | | Owner of the table |
| TABLE_NAME | VARCHAR2(128) | | Name of the table |
| CONSTRAINT | VARCHAR2(128) | | Integrity constraint that was violated |

# FLASHBACK_TRANSACTION_QUERY

FLASHBACK_TRANSACTION_QUERY displays information about all flashback transaction queries in the database. The database must have at least minimal supplemental logging enabled to avoid unpredictable behavior.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| XID | RAW(8) | | Transaction identifier |
| START_SCN | NUMBER | | Transaction start system change number (SCN) |
| START_TIMESTAMP | DATE | | Transaction start timestamp |
| COMMIT_SCN | NUMBER | | Transaction commit system change number; NULL for active transactions |
| COMMIT_TIMESTAMP | DATE | | Transaction commit timestamp; NULL for active transactions |
| LOGON_USER | VARCHAR2(30) | | Logon user for the transaction |
| UNDO_CHANGE# | NUMBER | | Undo system change number (1 or higher) |
| OPERATION | VARCHAR2(32) | | Forward-going DML operation performed by the transaction:<br>• D - Delete<br>• I - Insert<br>• U - Update<br>• B<br>• UNKNOWN |
| TABLE_NAME | VARCHAR2(256) | | Name of the table to which the DML applies |
| TABLE_OWNER | VARCHAR2(32) | | Owner of the table to which the DML applies |
| ROW_ID | VARCHAR2(19) | | Rowid of the row that was modified by the DML |
| UNDO_SQL | VARCHAR2(4000) | | SQL to undo the DML indicated by OPERATION |

**See Also:**

*Oracle Database Utilities* for information on how to enable minimal supplemental logging

## GLOBAL_CONTEXT

GLOBAL_CONTEXT displays the values of global context attributes, which are accessible for the current session, based on the CLIENT_IDENTIFIER value. This view is similar to SESSION_CONTEXT, which lists the values of session (or local) context attributes set under the current session.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAMESPACE | VARCHAR2(31) | | Namespace of the globally accessible context |
| ATTRIBUTE | VARCHAR2(31) | | Attribute of the globally accessible context |
| VALUE | VARCHAR2(4000) | | Value of the attribute of the globally accessible context |
| USERNAME | VARCHAR2(31) | | Username for which globally accessible context value is applicable |
| CLIENT_IDENTIFIER | VARCHAR2(65) | | Client identifier of the globally accessible context |

**See Also:**

- "SESSION_CONTEXT"
- *Oracle Database Security Guide* for more information about using global application contexts

## GLOBAL_NAME

GLOBAL_NAME contains one row that displays the global name of the current database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GLOBAL_NAME | VARCHAR2(4000) | | Global name of the database |

## HS_ALL_CAPS

HS_ALL_CAPS contains information about all of the capabilities (that is, features) associated with non-Oracle (FDS) data stores.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAP_NUMBER | NUMBER | | Capability number |
| CONTEXT | NUMBER | | Context in which this capability is applicable |
| TRANSLATION | VARCHAR2(255) | | Valid for functions; contains translation to FDS dialect |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ADDITIONAL_INFO | NUMBER | | Flag for internal use |
| FDS_CLASS_NAME | VARCHAR2(30) | | Name of the FDS Class |
| FDS_INST_NAME | VARCHAR2(30) | | Name of the FDS instance |

## HS_ALL_DD

HS_ALL_DD contains data dictionary information about non-Oracle (FDS) data stores.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DD_TABLE_NAME | VARCHAR2(128) | | Data dictionary table name |
| TRANSLATION_TYPE | CHAR(1) | | T = Translation, M = Mimic |
| TRANSLATION_TEXT | VARCHAR2(4000) | | SQL statement containing the mapping |
| FDS_CLASS_NAME | VARCHAR2(30) | | Name of the FDS Class |
| FDS_INST_NAME | VARCHAR2(30) | | Name of the FDS instance |
| DD_TABLE_DESC | VARCHAR2(255) | | Description of the Oracle data dictionary table |

## HS_ALL_INITS

HS_ALL_INITS contains initialization parameter information about non-Oracle (FDS) data stores.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INIT_VALUE_NAME | VARCHAR2(64) | | Name of the initialization parameter |
| INIT_VALUE | VARCHAR2(255) | | Value of the initialization parameter |
| INIT_VALUE_TYPE | VARCHAR2(1) | | Environment variable (T or F). T means this is an environment variable; F means do not set as an environment variable |
| FDS_CLASS_NAME | VARCHAR2(30) | | Name of the FDS Class |
| FDS_INST_NAME | VARCHAR2(30) | | Name of the FDS instance |

## HS_BASE_CAPS

HS_BASE_CAPS contains information about base capability (that is, base features) of the non-Oracle (FDS) data store.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAP_NUMBER | NUMBER | NOT NULL | Capability number |
| CAP_DESCRIPTION | VARCHAR2(255) | | Description of the capability |

# HS_BASE_DD

HS_BASE_DD displays information from the base data dictionary translation table.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DD_TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the Oracle data dictionary table |
| DD_TABLE_DESC | VARCHAR2(255) | | Description of the Oracle data dictionary table |
| DD_TABLE_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_CLASS_CAPS

HS_CLASS_CAPS contains information about the class-specific (driver) capabilities belonging to the non-Oracle (FDS) data store.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAP_NUMBER | NUMBER | NOT NULL | Capability number |
| CAP_DESCRIPTION | VARCHAR2(255) | | Capability description |
| CONTEXT | NUMBER | | Flag indicating the context in which the capability is enabled |
| TRANSLATION | VARCHAR2(255) | | Valid for functions; contains translation to FDS dialect |
| ADDITIONAL_INFO | NUMBER | | Additional flags for internal use |
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS Class |
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_CLASS_DD

HS_CLASS_DD displays information from the non-Oracle data store (FDS) class-specific data dictionary translations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DD_TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the Oracle data dictionary table |
| DD_TABLE_DESC | VARCHAR2(255) | | Description of the Oracle data dictionary table |
| TRANSLATION_TYPE | CHAR(1) | NOT NULL | T = Translation, M = Mimic |
| TRANSLATION_TEXT | VARCHAR2(4000) | | SQL statement containing the mapping |
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS Class |
| DD_TABLE_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_CLASS_INIT

HS_CLASS_INIT displays information about the non-Oracle (FDS) class-specific initialization parameters.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INIT_VALUE_NAME | VARCHAR2(64) | NOT NULL | Name of the initialization parameter |
| INIT_VALUE | VARCHAR2(255) | NOT NULL | Value of the initialization parameter |
| INIT_VALUE_TYPE | VARCHAR2(1) | NOT NULL | Environment variable (T or F). T means this is an environment variable; F means do not set as an environment variable |
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS Class |
| FDS_CLASS_INIT_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_FDS_CLASS

HS_FDS_CLASS contains information about legal non-Oracle (FDS) classes.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS class (for example, ODBC, DB2) |
| FDS_CLASS _COMMENTS | VARCHAR2(255) | | Text description of the non-Oracle class |
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_FDS_INST

HS_FDS_INST contains information about non-Oracle (FDS) instances.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FDS_INST_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS instance |
| FDS_INST_COMMENTS | VARCHAR2(255) | | Text description of the non-Oracle instance |
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS class |
| FDS_INST_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_INST_CAPS

HS_INST_CAPS contains information about instance-specific capabilities (that is, features).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAP_NUMBER | NUMBER | NOT NULL | Capability number |
| CAP_DESCRIPTION | VARCHAR2(255) | | Capability description |
| CONTEXT | NUMBER | | Context in which this capability is applicable |
| TRANSLATION | VARCHAR2(255) | | Valid for functions; contains translation to FDS dialect |
| ADDITIONAL_INFO | NUMBER | | Additional flags for internal use |
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS class (for example, ODBC, DB2) |
| FDS_INST_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS instance |
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |
| FDS_INST_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_INST_DD

HS_INST_DD displays information from the non-Oracle (FDS) instance-specific data dictionary translations.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DD_TABLE_NAME | VARCHAR2(128) | NOT NULL | Name of the Oracle data dictionary table |
| DD_TABLE_DESC | VARCHAR2(255) | | Description of the Oracle data dictionary table |
| TRANSLATION_TYPE | CHAR(1) | NOT NULL | T = Translation, M = Mimic |
| TRANSLATION_TEXT | VARCHAR2(4000) | | SQL statement containing the mapping |
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS class (for example, ODBC, DB2) |
| FDS_INST_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS instance |
| DD_TABLE_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |
| FDS_INST_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# HS_INST_INIT

HS_INST_INIT contains information about the non-Oracle (FDS) instance-specific initialization parameters.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INIT_VALUE_NAME | VARCHAR2(64) | NOT NULL | Name of the initialization parameter |
| INIT_VALUE | VARCHAR2(255) | NOT NULL | Value of the initialization parameter |
| INIT_VALUE_TYPE | VARCHAR2(1) | NOT NULL | Environment variable (T or F). T means this is an environment variable; F means do not set as an environment variable |
| FDS_CLASS_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS class (for example: ODBC, DB2) |
| FDS_INST_NAME | VARCHAR2(30) | NOT NULL | Name of the FDS instance |
| FDS_INST_INIT_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |
| FDS_CLASS_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |
| FDS_INST_ID | NUMBER | NOT NULL | Sequence: a counter that is incremented for every row inserted (used internally) |

# IDEPTREE

This view, created by utldtree.sql, lists the indented dependency tree. It is a pre-sorted, pretty-print version of DEPTREE.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NESTED_LEVEL | NUMBER | | Nesting level in the dependency tree |
| TYPE | VARCHAR2(23) | | Object type |
| OWNER | VARCHAR2(28) | | Object schema |
| NAME | VARCHAR2(1002) | | Object name |

# IND

IND is a synonym for USER_INDEXES.

**See Also:**

"USER_INDEXES"

# INDEX_HISTOGRAM

INDEX_HISTOGRAM contains information from the ANALYZE INDEX ... VALIDATE STRUCTURE statement.

**Note:**

The ANALYZE INDEX ... VALIDATE STRUCTURE OFFLINE statement must be used to collect statistics.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REPEAT_COUNT | NUMBER | | Number of times that one or more index keys is repeated in the table |
| KEYS_WITH_REPEAT_COUNT | NUMBER | | Number of index keys that are repeated that many times |

# INDEX_STATS

INDEX_STATS stores information from the last ANALYZE INDEX ... VALIDATE STRUCTURE statement.

**Note:**

The ANALYZE INDEX ... VALIDATE STRUCTURE OFFLINE statement must be used in order to collect statistics

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HEIGHT | NUMBER | | Height of the B-Tree |
| BLOCKS | NUMBER | NOT NULL | Blocks allocated to the segment |
| NAME | VARCHAR2(128) | NOT NULL | Name of the index |
| PARTITION_NAME | VARCHAR2(128) | | Name of the partition of the index which was analyzed. If the index is not partitioned, NULL is returned. |
| LF_ROWS | NUMBER | | Number of leaf rows (values in the index) |
| LF_BLKS | NUMBER | | Number of leaf blocks in the B-Tree |
| LF_ROWS_LEN | NUMBER | | Sum of the lengths of all the leaf rows |
| LF_BLK_LEN | NUMBER | | Usable space in a leaf block |
| BR_ROWS | NUMBER | | Number of branch rows in the B-Tree |
| BR_BLKS | NUMBER | | Number of branch blocks in the B-Tree |
| BR_ROWS_LEN | NUMBER | | Sum of the lengths of all the branch blocks in the B-Tree |
| BR_BLK_LEN | NUMBER | | Usable space in a branch block |
| DEL_LF_ROWS | NUMBER | | Number of deleted leaf rows in the index |
| DEL_LF_ROWS_LEN | NUMBER | | Total length of all deleted rows in the index |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DISTINCT_KEYS | NUMBER | | Number of distinct keys in the index (may include rows that have been deleted) |
| MOST_REPEATED_KEY | NUMBER | | How many times the most repeated key is repeated (may include rows that have been deleted) |
| BTREE_SPACE | NUMBER | | Total space currently allocated in the B-Tree |
| USED_SPACE | NUMBER | | Total space that is currently being used in the B-Tree |
| PCT_USED | NUMBER | | Percent of space allocated in the B-Tree that is being used |
| ROWS_PER_KEY | NUMBER | | Average number of rows per distinct key (this figure is calculated without consideration of deleted rows) |
| BLKS_GETS_PER_ACCESS | NUMBER | | Expected number of consistent mode block reads per row, assuming that a randomly chosen row is accessed using the index. Used to calculate the number of consistent reads that will occur during an index scan. |
| PRE_ROWS | NUMBER | | Number of prefix rows (values in the index) |
| PRE_ROWS_LEN | NUMBER | | Sum of lengths of all prefix rows |
| OPT_CMPR_COUNT | NUMBER | | Optimal index compression length |
| OPT_CMPR_PCTSAVE | NUMBER | | Corresponding space savings after an ANALYZE |

# LOGSTDBY_UNSUPPORTED_TABLES

LOGSTDBY_UNSUPPORTED_TABLES is a synonym for DBA_LOGSTDBY_UNSUPPORTED_TABLE.

**See Also:**

"DBA_LOGSTDBY_UNSUPPORTED_TABLE"

# MAP_OBJECT

MAP_OBJECT is a global temporary table that displays the hierarchical arrangement of storage containers for objects. Each row in the table represents a level in the hierarchy.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_NAME | VARCHAR2(2000) | | Name of the object |
| OBJECT_OWNER | VARCHAR2(2000) | | Owner of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_TYPE | VARCHAR2(2000) | | Object type |
| FILE_MAP_IDX | NUMBER | | File index (corresponds to FILE_MAP_IDX in V$MAP_FILE) |
| DEPTH | NUMBER | | Element depth within the I/O stack |
| ELEM_IDX | NUMBER | | Index corresponding to the element |
| CU_SIZE | NUMBER | | Contiguous set of logical blocks of the file (in HKB units) that is resident contiguously on the element |
| STRIDE | NUMBER | | Number of HKB between contiguous units (CU) in the file that are contiguous on this element. Used in RAID5 and striped files. |
| NUM_CU | NUMBER | | Number of contiguous units that are adjacent to each other on this element that are separated by STRIDE HKB in the file. In RAID5, the number of contiguous units also include the parity stripes. |
| ELEM_OFFSET | NUMBER | | Element offset (in HKB units) |
| FILE_OFFSET | NUMBER | | Offset (in HKB units) from the start of the file to the first byte of the contiguous units |
| DATA_TYPE | VARCHAR2(2000) | | Data type (DATA, PARITY, or DATA AND PARITY) |
| PARITY_POS | NUMBER | | Position of the parity. Only for RAID5. This field is needed to distinguish the parity from the data part. |
| PARITY_PERIOD | NUMBER | | Parity period. Only for RAID5. |

# NLS_DATABASE_PARAMETERS

NLS_DATABASE_PARAMETERS lists permanent NLS parameters of the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETER | VARCHAR2(128) | | Parameter name |
| VALUE | VARCHAR2(64) | | Parameter value |

# NLS_INSTANCE_PARAMETERS

NLS_INSTANCE_PARAMETERS lists NLS parameters of the instance.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETER | VARCHAR2(30) | | Parameter name |
| VALUE | VARCHAR2(64) | | Parameter value |

## NLS_SESSION_PARAMETERS

NLS_SESSION_PARAMETERS lists NLS parameters of the user session.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PARAMETER | VARCHAR2(30) | | Parameter name |
| VALUE | VARCHAR2(64) | | Parameter value |

## OBJ

OBJ is a synonym for USER_OBJECTS.

**See Also:**

"USER_OBJECTS"

## PATH_VIEW

PATH_VIEW contains one row for each unique path to access a resource in the Oracle XML DB repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PATH | VARCHAR2(1024) | | An (absolute) path to repository resource RES |
| RES | XMLTYPE(XMLSchema "http://xmlns.oracle.com/xdb/XDBResource.xsd" Element "Resource") | | The resource referred to by the PATH column |
| LINK | XMLTYPE | | Link property |
| RESID | RAW(16) | | Resource OID |

**See Also:**

*Oracle XML DB Developer's Guide* for information about using this view

## PDB_ALERTS

PDB_ALERTS contains descriptions of reasons for PDB alerts.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TIME | TIMESTAMP(6) | NOT NULL | Time when the violation happened |
| NAME | VARCHAR2(30) | NOT NULL | A name of a PDB or non-CDB to which this record applies |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| CAUSE_NO | NUMBER | NOT NULL | Number identifying a specific reason for a PDB alert |
| TYPE_NO | NUMBER | NOT NULL | Type of the violation |
| ERROR | NUMBER | | Oracle error, if any, for this violation |
| LINE | NUMBER | NOT NULL | Line number for the violation message |
| MESSAGE | VARCHAR2(4000) | NOT NULL | Description of the violation |
| STATUS | NUMBER | | Status of the violation |
| ACTION | VARCHAR2(4000) | | Actions to take to resolve the violations |

# PDB_PLUG_IN_VIOLATIONS

PDB_PLUG_IN_VIOLATIONS displays information about incompatibilities between a PDB and the CDB to which it belongs. This view is also used to display information generated by executing DBMS_PDB.CHECK_PLUG_COMPATIBILITY.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TIME | TIMESTAMP(6) | NOT NULL | Time when a violation described by this row was discovered |
| NAME | VARCHAR2(30) | NOT NULL | The name of an existing PDB or a PDB intended to be created (if a row was entered as a result of running DBMS_PDB.CHECK_PLUG_COMPATIBILITY) |
| CAUSE | VARCHAR2(22) | NOT NULL | Attribute which was being checked |
| TYPE | VARCHAR2(9) | NOT NULL | ERROR or WARNING |
| ERROR_NUMBER | NUMBER | | Oracle error number, if any, encountered during a check |
| LINE | NUMBER | NOT NULL | Used to differentiate between violations which share a cause |
| MESSAGE | VARCHAR2(4000) | NOT NULL | Description of a violation |
| STATUS | VARCHAR2(9) | | PENDING, RESOLVED, or IGNORE |
| ACTION | VARCHAR2(4000) | | Description of an action to take to correct the violation |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_PDB package, which provides an interface for examining and manipulating data about PDBs

# PLAN_TABLE

PLAN_TABLE is automatically created as a global temporary table to hold the output of an EXPLAIN PLAN statement for all users. PLAN_TABLE is the default sample output table into which the EXPLAIN PLAN statement inserts rows describing execution plans.

While a PLAN_TABLE table is automatically set up for each user, you can use the SQL script utlxplan.sql to manually create a local PLAN_TABLE in your schema.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| STATEMENT_ID | VARCHAR2(30) | | Value of the optional STATEMENT_ID parameter specified in the EXPLAIN PLAN statement |
| PLAN_ID | NUMBER | | Unique identifier of a plan in the database |
| TIMESTAMP | DATE | | Date and time when the EXPLAIN PLAN statement was generated |
| REMARKS | VARCHAR2(4000) | | Any comment (of up to 4000 bytes) you want to associate with each step of the explained plan. This column is used to indicate whether an outline or SQL Profile was used for the query.<br>If you need to add or change a remark on any row of the PLAN_TABLE, then use the UPDATE statement to modify the rows of the PLAN_TABLE. |
| OPERATION | VARCHAR2(30) | | Name of the internal operation performed in this step. In the first row generated for a statement, the column contains one of the following values:<br>• DELETE STATEMENT<br>• INSERT STATEMENT<br>• SELECT STATEMENT<br>• UPDATE STATEMENT |
| OPTIONS | VARCHAR2(255) | | A variation on the operation described in the OPERATION column |
| OBJECT_NODE | VARCHAR2(128) | | Name of the database link used to reference the object (a table name or view name). For local queries using parallel execution, this column describes the order in which output from operations is consumed. |
| OBJECT_OWNER | VARCHAR2(128) | | Owner of the table or index |
| OBJECT_NAME | VARCHAR2(128) | | Name of the table or index |
| OBJECT_ALIAS | VARCHAR2(261) | | Unique alias of a table or view in a SQL statement. For indexes, it is the object alias of the underlying table. |
| OBJECT_INSTANCE | NUMBER(38) | | Number corresponding to the ordinal position of the object as it appears in the original statement. The numbering proceeds from left to right, outer to inner with respect to the original statement |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | text. View expansion results in unpredictable numbers. |
| OBJECT_TYPE | VARCHAR2(30) | | Modifier that provides descriptive information about the object; for example, NON-UNIQUE for indexes |
| OPTIMIZER | VARCHAR2(255) | | Current mode of the optimizer |
| SEARCH_COLUMNS | NUMBER | | Not currently used |
| ID | NUMBER(38) | | A number assigned to each step in the execution plan |
| PARENT_ID | NUMBER(38) | | ID of the next execution step that operates on the output of the ID step |
| DEPTH | NUMBER(38) | | Depth of the operation in the row source tree that the plan represents. The value can be used for indenting the rows in a plan table report. |
| POSITION | NUMBER(38) | | For the first row of output, this indicates the optimizer's estimated cost of executing the statement. For the other rows, it indicates the position relative to the other children of the same parent. |
| COST | NUMBER(38) | | Cost of the operation as estimated by the optimizer's query approach. Cost is not determined for table access operations. The value of this column does not have any particular unit of measurement; it is merely a weighted value used to compare costs of execution plans. The value of this column is a function of the CPU_COST and IO_COST columns. |
| CARDINALITY | NUMBER(38) | | Estimate by the query optimization approach of the number of rows accessed by the operation |
| BYTES | NUMBER(38) | | Estimate by the query optimization approach of the number of bytes accessed by the operation |
| OTHER_TAG | VARCHAR2(255) | | Describes the contents of the OTHER column:<br>• SERIAL - Serial execution. Currently, SQL is not loaded in the OTHER column for this case.<br>• SERIAL_FROM_REMOTE - Serial execution at a remote site.<br>• PARALLEL_FROM_SERIAL - Serial execution. Output of step is partitioned or broadcast to parallel execution servers.<br>• PARALLEL_TO_SERIAL - Parallel execution. Output of step is returned to serial query coordinator (QC) process.<br>• PARALLEL_TO_PARALLEL - Parallel execution. Output of step is repartitioned to second set of parallel execution servers.<br>• PARALLEL_COMBINED_WITH_PARENT - Parallel execution; Output of step goes to next |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | step in same parallel process. No interprocess communication to parent.<br>• PARALLEL_COMBINED_WITH_CHILD - Parallel execution. Input of step comes from prior step in same parallel process. No interprocess communication from child. |
| PARTITION_START | VARCHAR2(255) | | Start partition of a range of accessed partitions:<br>• *number* - Start partition has been identified by the SQL compiler, and its partition number is given by *number*<br>• KEY - Start partition will be identified at run time from partitioning key values<br>• ROW REMOVE_LOCATION - Start partition (same as the stop partition) will be computed at run time from the location of each record being retrieved. The record location is obtained by a user or from a global index.<br>• INVALID - Range of accessed partitions is empty |
| PARTITION_STOP | VARCHAR2(255) | | Stop partition of a range of accessed partitions:<br>• *number* - Stop partition has been identified by the SQL compiler, and its partition number is given by *number*<br>• KEY - Stop partition will be identified at run time from partitioning key values<br>• ROW REMOVE_LOCATION - Stop partition (same as the start partition) will be computed at run time from the location of each record being retrieved. The record location is obtained by a user or from a global index.<br>• INVALID - Range of accessed partitions is empty |
| PARTITION_ID | NUMBER(38) | | Step that has computed the pair of values of the PARTITION_START and PARTITION_STOP columns |
| OTHER | LONG | | Other information that is specific to the execution step that a user might find useful (see the OTHER_TAG column) |
| OTHER_XML | CLOB | | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML since multiple pieces of information can be stored there. This includes:<br>• Name of the schema against which the query was parsed<br>• Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • Indication of whether or not dynamic statistics were used to produce the plan<br>• The outline data, a set of optimizer hints that can be used to regenerate the same plan |
| DISTRIBUTION | VARCHAR2(30) | | Method used to distribute rows from producer query servers to consumer query servers<br>**See Also:** *Oracle Database Data Warehousing Guide* for more information about consumer and producer query servers |
| CPU_COST | NUMBER(38) | | CPU cost of the operation as estimated by the query optimizer's approach. The value of this column is proportional to the number of machine cycles required for the operation. For statements that use the rule-based approach, this column is NULL. |
| IO_COST | NUMBER(38) | | I/O cost of the operation as estimated by the query optimizer's approach. The value of this column is proportional to the number of data blocks read by the operation. For statements that use the rule-based approach, this column is NULL. |
| TEMP_SPACE | NUMBER(38) | | Temporary space (in bytes) used by the operation as estimated by the query optimizer's approach. For statements that use the rule-based approach, or for operations that do not use any temporary space, this column is NULL. |
| ACCESS_PREDICATES | VARCHAR2(4000) | | Predicates used to locate rows in an access structure. For example, start or stop predicates for an index range scan. |
| FILTER_PREDICATES | VARCHAR2(4000) | | Predicates used to filter rows before producing them |
| PROJECTION | VARCHAR2(4000) | | Expressions produced by the operation |
| TIME | NUMBER(38) | | Elapsed time (in seconds) of the operation as estimated by query optimization. For statements that use the rule-based approach, this column is NULL. |
| QBLOCK_NAME | VARCHAR2(30) | | Name of the query block (either system-generated or defined by the user with the QB_NAME hint) |

# PLUGGABLE_SET_CHECK

PLUGGABLE_SET_CHECK contains pluggable set checks.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJ1_OWNER | VARCHAR2(128) | | Owner of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJ1_NAME | VARCHAR2(128) | | Object 1 |
| OBJ1_SUBNAME | VARCHAR2(128) | | SubObject1Name |
| OBJ1_TYPE | VARCHAR2(18) | | Object Type |
| TS1_NAME | VARCHAR2(30) | | Tablespace containing Object 1 |
| OBJ2_NAME | VARCHAR2(128) | | Object Name |
| OBJ2_SUBNAME | VARCHAR2(128) | | SubObject2Name |
| OBJ2_TYPE | VARCHAR2(18) | | Object Type |
| OBJ2_OWNER | VARCHAR2(128) | | Object owner of second object |
| TS2_NAME | VARCHAR2(30) | | Tablespace containing Object 1 |
| CONSTRAINT_NAME | VARCHAR2(128) | | Name of dependent constraint |
| REASON | VARCHAR2(86) | | Reason for Pluggable check violation |
| MESG_ID | NUMBER | | The message ID |

# PRODUCT_COMPONENT_VERSION

PRODUCT_COMPONENT_VERSION contains version and status information for component products.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PRODUCT | VARCHAR2(80) | | Product name |
| VERSION | VARCHAR2(80) | | Version number |
| STATUS | VARCHAR2(80) | | Status of release |

# PROXY_USERS

PROXY_USERS describes the list of proxy users and the clients on whose behalf they can act.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PROXY | VARCHAR2(128) | NOT NULL | Name of a proxy user |
| CLIENT | VARCHAR2(128) | NOT NULL | Name of the client user who the proxy user can act as |
| AUTHENTICATION | VARCHAR2(3) | | Indicates whether the proxy is required to supply the client's authentication credentials (YES) or not (NO) |
| FLAGS | VARCHAR2(35) | | Flags associated with the proxy/client pair:<br>• PROXY MAY ACTIVATE ALL CLIENT ROLES<br>• NO CLIENT ROLES MAY BE ACTIVATED<br>• PROXY MAY ACTIVATE ROLE |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • PROXY MAY ACTIVATE ALL CLIENT ROLES<br>• PROXY MAY NOT ACTIVATE ROLE |

# PSTUBTBL

This table contains information on stubs generated by the PSTUB utility so that an Oracle Forms 3.0 client can call stored procedures in Oracle Database.

**Note:**

The contents of this table are intended only for use by the PSTUB utility.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | | Schema part of the identifier of a stored procedure |
| DBNAME | VARCHAR2(128) | | Database link part of the identifier of a stored procedure |
| LUN | VARCHAR2(128) | | Library unit name part of the identifier of a stored procedure |
| LUTYPE | VARCHAR2(3) | | Type of the stored procedure |
| LINENO | NUMBER | | Line number of the stub |
| LINE | VARCHAR2(1800) | | Text of the stub |

# PUBLIC_DEPENDENCY

PUBLIC_DEPENDENCY lists dependencies to and from objects, by object number.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_ID | NUMBER | NOT NULL | Object number |
| REFERENCED_OBJECT_ID | NUMBER | NOT NULL | Referenced object (the parent object) |

# PUBLICSYN

PUBLICSYN contains information on public synonyms.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| SNAME | VARCHAR2(128) | | Name of the synonym |
| CREATOR | VARCHAR2(128) | | Owner of the synonym |
| TNAME | VARCHAR2(128) | | Table of which this is a synonym |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| DATABASE | VARCHAR2(128) | | Database in which the table resides |
| TABTYPE | VARCHAR2(10) | | Type of table |

# QUEUE_PRIVILEGES

QUEUE_PRIVILEGES shows all Advanced Queuing object privileges granted to the session.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| GRANTEE | VARCHAR2(128) | NOT NULL | Name of the user or role to whom access was granted |
| OWNER | VARCHAR2(128) | NOT NULL | Owner of the object |
| NAME | VARCHAR2(128) | NOT NULL | Name of the object |
| GRANTOR | VARCHAR2(128) | NOT NULL | Name of the user who performed the grant |
| ENQUEUE_PRIVILEGE | NUMBER | | Permission to ENQUEUE to the queue |
| DEQUEUE_PRIVILEGE | NUMBER | | Permission to DEQUEUE from the queue |

# RECYCLEBIN

RECYCLEBIN is a synonym for USER_RECYCLEBIN.

**See Also:**

"USER_RECYCLEBIN"

# REDACTION_COLUMNS

REDACTION_COLUMNS describes all redacted columns in the database, giving the owner of the table or view within which the column resides, the object name, the column name, the type of redaction function, the parameters to the redaction function (if any), and an optional description of the redaction policy.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object that is redacted |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object that is redacted |
| COLUMN_NAME | VARCHAR2(128) | NOT NULL | Name of the column that is redacted |
| FUNCTION_TYPE | VARCHAR2(27) | | Redaction function for this column |
| FUNCTION_PARAMETERS | VARCHAR2(1000) | | Redaction function_parameters for this column |
| REGEXP_PATTERN | VARCHAR2(512) | | Regular expression pattern to search for |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| REGEXP_REPLACE_STRING | VARCHAR2(4000) | | Replacement string (up to 4000 characters in length) with up to 500 back-references to subexpressions in the form \n, (where n is a number from 1 to 9) |
| REGEXP_POSITION | NUMBER | | Integer counting from 1, giving the position where the search should begin |
| REGEXP_OCCURRENCE | NUMBER | | Either 0 (to replace all occurrences of the match), or a positive integer n (to replace the nth occurrence of the match) |
| REGEXP_MATCH_PARAMETER | VARCHAR2(10) | | To change the default matching behavior, possible values are a combination of i, c, n, m, and x. See the documentation of the match_parameter in the REGEXP_REPLACE section of the *Oracle Database SQL Language Reference*. |
| COLUMN_DESCRIPTION | VARCHAR2(4000) | | User-provided description of the redaction function that is performed on the column. For example, for a Social Security Number column, the description might be: "redact SSN to XXX-XX-(last 4 digits)". |

**See Also:**

*Oracle Database Advanced Security Guide* for more information about Oracle Data Redaction

# REDACTION_POLICIES

REDACTION_POLICIES displays all redaction policies in the database.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_OWNER | VARCHAR2(128) | NOT NULL | Owner of the object with the policy |
| OBJECT_NAME | VARCHAR2(128) | NOT NULL | Name of the object with the policy |
| POLICY_NAME | VARCHAR2(128) | NOT NULL | Name of the policy |
| EXPRESSION | VARCHAR2(4000) | NOT NULL | Expression for this policy |
| ENABLE | VARCHAR2(7) | | Indicates whether the policy is enabled (YES) or not (NO) |
| POLICY_DESCRIPTION | VARCHAR2(4000) | | Description of the policy |

**See Also:**

*Oracle Database Advanced Security Guide* for more information about Oracle Data Redaction

# REDACTION_VALUES_FOR_TYPE_FULL

REDACTION_VALUES_FOR_TYPE_FULL shows all of the current values for full redaction.

For example, if a redaction policy is applied to a column of type BINARY_DOUBLE and the redaction type is full redaction, that column will be redacted with the value shown in the BINARY_DOUBLE_VALUE column of this view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NUMBER_VALUE | NUMBER | NOT NULL | Redaction result for full redaction on NUMBER columns |
| BINARY_FLOAT_VALUE | BINARY_FLOAT | NOT NULL | Redaction result for full redaction on BINARY_FLOAT columns |
| BINARY_DOUBLE_VALUE | BINARY_DOUBLE | NOT NULL | Redaction result for full redaction on BINARY_DOUBLE columns |
| CHAR_VALUE | VARCHAR2(1) | | Redaction result for full redaction on CHAR columns |
| VARCHAR_VALUE | VARCHAR2(1) | | Redaction result for full redaction on VARCHAR2 columns |
| NCHAR_VALUE | NCHAR(1) | | Redaction result for full redaction on NCHAR columns |
| NVARCHAR_VALUE | NVARCHAR2(1) | | Redaction result for full redaction on NVARCHAR2 columns |
| DATE_VALUE | DATE | NOT NULL | Redaction result for full redaction on DATE columns |
| TIMESTAMP_VALUE | TIMESTAMP(6) | NOT NULL | Redaction result for full redaction on TIMESTAMP columns |
| TIMESTAMP_WITH_TIME_ZONE_VALUE | TIMESTAMP(6) WITH TIME ZONE | NOT NULL | Redaction result for full redaction on TIMESTAMP WITH TIME ZONE columns |
| BLOB_VALUE | BLOB | | Redaction result for full redaction on BLOB columns |
| CLOB_VALUE | CLOB | | Redaction result for full redaction on CLOB columns |
| NCLOB_VALUE | NCLOB | | Redaction result for full redaction on NCLOB columns |

**See Also:**

*Oracle Database Advanced Security Guide* olink:DBSEG594for more information about Oracle Data Redaction

# REPORT_COMPONENTS

REPORT_COMPONENTS displays metadata about different database components offering reports in XML, HTML, or Text formats. Reports are first generated in XML and can then be translated into HTML or Text formats, for supported report types. Each component generates one or more reports containing different types of content. You can request reports using the component's own PL/SQL interface (for example, DBMS_SQLTUNE for SQL Tuning Advisor).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPONENT_ID | NUMBER | NOT NULL | ID number of the database component building the report |
| COMPONENT_NAME | VARCHAR2(128) | NOT NULL | Name of the database component building the report (for example, sqltune for SQL Tuning Advisor) |
| COMPONENT_DESCRIPTION | VARCHAR2(256) | | Component description |
| REPORT_ID | NUMBER | NOT NULL | ID number of the report type |
| REPORT_NAME | VARCHAR2(128) | NOT NULL | Name of the report type |
| REPORT_DESCRIPTION | VARCHAR2(256) | | Description of the report type |
| SCHEMA_FILENAME | VARCHAR2(500) | | Filename of the XML schema for the report (optional) |
| SCHEMA_DATA | XMLTYPE | | XML schema for this report (optional) |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SQLTUNE package

# REPORT_FILES

REPORT_FILES displays data for all of the XML schema and XSLT files associated with reports for a given component.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILENAME | VARCHAR2(500) | NOT NULL | Name of the XSLT/XML schema file |
| DATA | XMLTYPE | | XSLT/XML schema data |

# REPORT_FORMATS

REPORT_FORMATS displays metadata about the different output formats supported for reports. Some reports are generated in XML only, in which case no data will appear in this view. When reports support HTML or Text formats (for example, the SQL Performance Analyzer reports), metadata about the format will appear in this view.

XML reports can be translated to another format with the DBMS_REPORT.FORMAT_REPORT procedure.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| COMPONENT_ID | NUMBER | NOT NULL | ID number of the database component building the report |
| COMPONENT_NAME | VARCHAR2(128) | NOT NULL | Name of the database component building the report (for example, sqltune for SQL Tuning Advisor) |
| REPORT_ID | NUMBER | NOT NULL | ID number of the report type |
| REPORT_NAME | VARCHAR2(128) | NOT NULL | Name of the report type |
| FORMAT_NAME | VARCHAR2(128) | NOT NULL | Name of the report format |
| DESCRIPTION | VARCHAR2(256) | | Description of the report format |
| TYPE | VARCHAR2(6) | | Format type:<br>• XSLT - Reports generated by applying an XSLT style sheet to XML data (for example, HTML reports)<br>• Text - Reports generated by first applying an XSLT style sheet to convert XML data to HTML, and then converting the HTML to formatted Text using the internal report HTML-to-text translation engine.<br>• Custom - Custom formats implemented natively by report clients |
| XSLT_FILENAME | VARCHAR2(500) | | Name of the XSLT used for this format (XSLT and Text format types only) |
| XSLT_DATA | XMLTYPE | | XSLT data (XSLT and Text format types only) |
| TEXT_LINESIZE | NUMBER | | Maximum line size of the formatted text report (Text format types only) |

# RESOURCE_COST

RESOURCE_COST lists the cost for each resource.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RESOURCE_NAME | VARCHAR2(32) | NOT NULL | Name of the resource |
| UNIT_COST | NUMBER | NOT NULL | Cost of the resource |

# RESOURCE_MAP

RESOURCE_MAP describes resources. This table can be used to map resource names to resource numbers.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RESOURCE# | NUMBER | NOT NULL | Numeric resource code |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TYPE# | NUMBER | NOT NULL | Numeric type code |
| NAME | VARCHAR2(32) | NOT NULL | Name of the resource |

## RESOURCE_VIEW

RESOURCE_VIEW contains one row for each resource in the Oracle XML DB repository.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RES | XMLTYPE(XMLSchema "http://xmlns.oracle.com/xdb/XDBResource.xsd" Element "Resource") | | A resource in the repository |
| ANY_PATH | VARCHAR2(4000) | | An (absolute) path to the resource |
| RESID | RAW(16) | | Resource OID, which is a unique handle to the resource |

**See Also:**

*Oracle XML DB Developer's Guide* for information about using this view

## ROLE_ROLE_PRIVS

ROLE_ROLE_PRIVS describes the roles granted to other roles. Information is provided only about roles to which the user has access.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROLE | VARCHAR2(128) | | Name of the role |
| GRANTED_ROLE | VARCHAR2(128) | | Role that was granted |
| ADMIN_OPTION | VARCHAR2(3) | | Signifies that the role was granted with ADMIN option |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the role was granted commonly (CONTAINER=ALL was used)<br>• NO if the role was granted locally (CONTAINER=ALL was not used) |

## ROLE_SYS_PRIVS

ROLE_SYS_PRIVS describes system privileges granted to roles. Information is provided only about roles to which the user has access.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROLE | VARCHAR2(128) | | Name of the role |
| PRIVILEGE | VARCHAR2(40) | | System privilege granted to the role |
| ADMIN_OPTION | VARCHAR2(3) | | Indicates whether the grant was with the ADMIN option (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |

# ROLE_TAB_PRIVS

ROLE_TAB_PRIVS describes table privileges granted to roles. Information is provided only about roles to which the user has access.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROLE | VARCHAR2(128) | | Name of the role |
| OWNER | VARCHAR2(128) | | Owner of the object |
| TABLE_NAME | VARCHAR2(128) | | Name of the object |
| COLUMN_NAME | VARCHAR2(128) | | Name of the column, if applicable |
| PRIVILEGE | VARCHAR2(40) | | Object privilege granted to the role |
| GRANTABLE | VARCHAR2(3) | | YES if the role was granted with ADMIN OPTION; otherwise NO |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |

# SCHEDULER_BATCH_ERRORS

SCHEDULER_BATCH_ERRORS displays the errors caused by each call in the batch after a Scheduler batch call (when the COMMIT_SEMANTICS argument has been set to ABSORB_ERRORS).

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ARRAY_INDEX | NUMBER | | Index of the job in the batch |
| OBJECT_TYPE | VARCHAR2(30) | | Object type:<br>• JOB<br>• LIGHTWEIGHT JOB<br>• UNKNOWN |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_NAME | VARCHAR2(100) | | Full name of the object (including schema) |
| ATTR_NAME | VARCHAR2(30) | | Name of the attribute being set (if this is a batch set attribute call); NULL otherwise |
| ERROR_CODE | NUMBER | | Top level error code |
| ERROR_MESSAGE | VARCHAR2(4000) | | Complete error stack |
| ADDITIONAL_INFO | VARCHAR2(4000) | | Additional information (currently unused) |

# SCHEMA_EXPORT_OBJECTS

SCHEMA_EXPORT_OBJECTS lists simple path names for some of the object types belonging to a Data Pump schema export, which is invoked using the SCHEMAS parameter on the expdp command. Users of the Data Pump Export and Import utilities can query this view to determine valid values for the EXCLUDE and INCLUDE parameters.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_PATH | VARCHAR2(200) | NOT NULL | Simple path name for the object type |
| COMMENTS | VARCHAR2(2000) | | Comment on the object type |
| NAMED | CHAR(1) | | Do objects of this type have names? If yes (Y), then the name can be specified in the optional name_clause on the EXCLUDE and INCLUDE parameters. |

**See Also:**

- "DATABASE_EXPORT_OBJECTS"
- "TABLE_EXPORT_OBJECTS"
- *Oracle Database Utilities* for more information on performing a full Data Pump export using the expdp command

# SEQ

SEQ is a synonym for USER_SEQUENCES.

**See Also:**

"USER_SEQUENCES"

## SESSION_CONTEXT

SESSION_CONTEXT describes the context attributes and their values set for the current session.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| NAMESPACE | VARCHAR2(30) | | Namespace that the active attribute is in |
| ATTRIBUTE | VARCHAR2(30) | | Name of the active attribute |
| VALUE | VARCHAR2(4000) | | Value of the active attribute |

## SESSION_PRIVS

SESSION_PRIVS describes the privileges that are currently available to the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PRIVILEGE | VARCHAR2(40) | NOT NULL | Name of the privilege |

## SESSION_ROLES

SESSION_ROLES describes the roles that are currently enabled to the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| ROLE | VARCHAR2(128) | NOT NULL | Name of the role |

## SOURCE_SIZE

Oracle accesses this view to create views about object size.

**See Also:**

"DBA_OBJECT_SIZE" and "USER_OBJECT_SIZE"

## STMT_AUDIT_OPTION_MAP

STMT_AUDIT_OPTION_MAP describes auditing option type codes. This table can be used to map auditing option type numbers to type names.

**Note:**

This view is populated in any Oracle Database where auditing is enabled, regardless of whether pre-Oracle Database 12*c* auditing or unified auditing is enabled for the database.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

**Note:**

The mapping explained in this view is valid for audit configuration from the DBA_STMT_AUDIT_OPTS view only, and such audit configurations can be made when unified auditing is not enabled.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OPTION# | NUMBER | NOT NULL | Numeric auditing option type code |
| NAME | VARCHAR2(40) | NOT NULL | Name of the type of auditing option |
| PROPERTY | NUMBER | NOT NULL | Property flag of the auditing option |

**See Also:**

"DBA_STMT_AUDIT_OPTS"

# SYN

SYN is a synonym for USER_SYNONYMS.

**See Also:**

"ALL_SYNONYMS"

# SYNONYMS

SYNONYMS is included for compatibility. Oracle recommends that you do not use this view.

# SYS_OBJECTS

SYS_OBJECTS maps object IDs to object types and segment data block addresses.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_TYPE | VARCHAR2(18) | | Type of the object |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_TYPE_ID | NUMBER | | Type ID of the object |
| SEGMENT_TYPE_ID | NUMBER | | Type of segment: TABLE, CLUSTER, INDEX, ROLLBACK, DEFERRED ROLLBACK, TEMPORARY, CACHE |
| OBJECT_ID | NUMBER | | Object identifier |
| HEADER_FILE | NUMBER | | ID of the file containing the segment header |
| HEADER_BLOCK | NUMBER | | ID of the block containing the segment header |
| TS_NUMBER | NUMBER | | The tablespace number |

## SYSCATALOG

SYSCATALOG is included for compatibility. Oracle recommends that you do not use this view.

## SYSFILES

SYSFILES is included for compatibility. Oracle recommends that you do not use this view.

## SYSSEGOBJ

SYSSEGOBJ is included for compatibility. Oracle recommends that you do not use this view.

## SYSTEM_PRIVILEGE_MAP

SYSTEM_PRIVILEGE_MAP describes privilege (auditing option) type codes. This table can be used to map privilege (auditing option) type numbers to type names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PRIVILEGE | NUMBER | NOT NULL | Numeric privilege (auditing option) type code |
| NAME | VARCHAR2(40) | NOT NULL | Name of the type of privilege (auditing option).<br>See *Oracle Database SQL Language Reference* for a list of valid system privileges. |
| PROPERTY | NUMBER | NOT NULL | Property flag of the privilege (auditing option):<br>• 0 - Indicates a privilege that can be granted with a SQL GRANT statement<br>• 1 - Indicates a privilege that can only be granted using a PL/SQL package |

## TAB

TAB is included for compatibility. Oracle recommends that you do not use this view.

# TABLE_EXPORT_OBJECTS

TABLE_EXPORT_OBJECTS lists simple path names for some of the object types belonging to a Data Pump schema export, which is invoked using the TABLES parameter on the expdp command. Users of the Data Pump Export and Import utilities can query this view to determine valid values for the EXCLUDE and INCLUDE parameters.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJECT_PATH | VARCHAR2(200) | NOT NULL | Simple path name for the object type |
| COMMENTS | VARCHAR2(2000) | | Comment on the object type |
| NAMED | CHAR(1) | | Do objects of this type have names? If yes (Y), then the name can be specified in the optional name_clause on the EXCLUDE and INCLUDE parameters. |

**See Also:**

- "DATABASE_EXPORT_OBJECTS"
- "SCHEMA_EXPORT_OBJECTS"
- *Oracle Database Utilities* for more information on performing a full Data Pump export using the expdp command

# TABLE_PRIVILEGE_MAP

TABLE_PRIVILEGE_MAP describes privilege (auditing option) type codes. This table can be used to map privilege (auditing option) type numbers to type names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PRIVILEGE | NUMBER | NOT NULL | Numeric privilege (auditing option) type code |
| NAME | VARCHAR2(40) | NOT NULL | Name of the type of privilege (auditing option) |

# TABQUOTAS

TABQUOTAS is included for compatibility. Oracle recommends that you do not use this view.

# TABS

TABS is a synonym for USER_TABLES.

**See Also:**

"USER_TABLES"

# TRUSTED_SERVERS

TRUSTED_SERVERS displays whether a server is trusted or untrusted.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TRUST | VARCHAR2(9) | | Trustedness of the server listed. Values can be TRUSTED or UNTRUSTED servers which are not listed in the NAME column have opposite trustedness. |
| NAME | VARCHAR2(128) | | Server name. Can be a specific server name or ALL for all servers. |

Table 6-1 shows examples of the values returned depending on the status of the servers.

***Table 6-1 TRUSTED_SERVERS Values***

| Condition (If . . . ) | TRUSTED column | NAME column |
|---|---|---|
| ... all servers are trusted | Trusted | ALL |
| ... no servers are trusted | Untrusted | ALL |
| ... all servers except DB1 are trusted | Untrusted | DB1 |
| ... all servers except DB1 are untrusted | Trusted | DB1 |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference*
- *Oracle Database Enterprise User Security Administrator's Guide*

# TS_PITR_CHECK

This view, created by `catpitr.sql`, provides information on any dependencies or restrictions that might prevent tablespace point-in-time recovery from proceeding. This view applies only to the tablespace point-in-time recovery feature.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJ1_OWNER | VARCHAR2(128) | | The owner of the object preventing tablespace point-in-time recovery. See the REASON column for details. |
| OBJ1_NAME | VARCHAR2(128) | | The name of the object preventing tablespace point-in-time recovery |
| OBJ1_SUBNAME | VARCHAR2(128) | | Subordinate to OBJ1_NAME |
| OBJ1_TYPE | VARCHAR2(16) | | The object type for the object preventing tablespace point-in-time recovery |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TS1_NAME | VARCHAR2(30) | | Name of the tablespace containing the object preventing tablespace point-in-time recovery |
| OBJ2_NAME | VARCHAR2(128) | | The name of a second object which may be preventing tablespace point-in-time recovery. If NULL, object 1 is the only object preventing recovery. |
| OBJ2_SUBNAME | VARCHAR2(128) | | Subordinate to OBJ2_NAME |
| OBJ2_TYPE | VARCHAR2(15) | | The object type for the second object (will be NULL if OBJ2_NAME is NULL) |
| OBJ2_OWNER | VARCHAR2(128) | | The owner of the second object (will be NULL if OBJ2_NAME is NULL) |
| TS2_NAME | VARCHAR2(30) | | Name of the tablespace containing second object which may be preventing tablespace point-in-time recovery (-1 indicates not applicable) |
| CONSTRAINT_NAME | VARCHAR2(128) | | Name of the constraint |
| REASON | VARCHAR2(81) | | Reason why tablespace point-in-time recovery cannot proceed |

**See Also:**

*Oracle Database Backup and Recovery User's Guide* for more information about tablespace point-in-time recovery

# TS_PITR_OBJECTS_TO_BE_DROPPED

TS_PITR_OBJECTS_TO_BE_DROPPED lists all objects lost as a result of performing tablespace point-in-time recovery. This view applies only to the tablespace point-in-time recovery feature.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OWNER | VARCHAR2(128) | NOT NULL | The owner of the object |
| NAME | VARCHAR2(128) | NOT NULL | The name of the object that will be lost as a result of undergoing tablespace point-in-time recovery |
| CREATION_TIME | DATE | NOT NULL | Creation timestamp of the object |
| TABLESPACE_NAME | VARCHAR2(30) | | Name of the tablespace containing the object |

**See Also:**

*Oracle Database Backup and Recovery User's Guide* for more information about tablespace point-in-time recovery

# UNI_PLUGGABLE_SET_CHECK

UNI_PLUGGABLE_SET_CHECK contains pluggable check information.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OBJ1_OWNER | VARCHAR2(128) | | Owner of object |
| OBJ1_NAME | VARCHAR2(128) | | Object 1 |
| OBJ1_SUBNAME | VARCHAR2(128) | | SubObject1Name |
| OBJ1_TYPE | VARCHAR2(18) | | Object Type |
| TS1_NAME | VARCHAR2(30) | | Tablespace containing Object 1 |
| OBJ2_NAME | VARCHAR2(128) | | Object Name |
| OBJ2_SUBNAME | VARCHAR2(128) | | SubObject2Name |
| OBJ2_TYPE | VARCHAR2(18) | | Object Type |
| OBJ2_OWNER | VARCHAR2(128) | | Object owner of second object |
| TS2_NAME | VARCHAR2(30) | | Tablespace containing Object 1 |
| CONSTRAINT_NAME | VARCHAR2(128) | | Name of dependent constraint |
| REASON | VARCHAR2(86) | | Reason for Pluggable check violation |
| MESG_ID | NUMBER | | The message ID |

# UNIFIED_AUDIT_TRAIL

When unified auditing is enabled in Oracle Database, the audit records are populated in this new audit trail. This view displays audit records in tabular form by retrieving the audit records from the audit trail.

Be aware that if the audit trail mode is QUEUED, then audit records are not written to disk until the in-memory queues are full. The following procedure explicitly flushes the queues to disk, so that you can see the audit trail records in the UNIFIED_AUDIT_TRAIL view:

```
EXEC SYS.DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL;
```

**Note:**

This view is populated only in an Oracle Database where unified auditing is enabled.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| AUDIT_TYPE | VARCHAR2(64) | | Type of auditing:<br>• Standard<br>• FineGrainedAudit<br>• XS<br>• Database Vault<br>• Label Security<br>• RMAN_AUDIT<br>• Datapump<br>• Direct path API |
| SESSIONID | NUMBER | | Audit session identifier |
| PROXY_SESSIONID | NUMBER | | Audit session identifier of proxying session |
| OS_USERNAME | VARCHAR2(30) | | Name of the operating system user for the database session |
| USERHOST | VARCHAR2(128) | | Name of the host machine from which the session was spawned |
| TERMINAL | VARCHAR2(30) | | The operating system terminal of the user session |
| INSTANCE_ID | NUMBER | | Instance number as specified in the initialization parameter file, init.ora |
| DBID | NUMBER | | Database identifier of the audited database |
| AUTHENTICATION_TYPE | VARCHAR2(1024) | | Type of authentication for the session user |
| DBUSERNAME | VARCHAR2(30) | | Database user name of the user whose actions were audited |
| DBPROXY_USERNAME | VARCHAR2(30) | | Proxying user name, in the case of proxy authentication |
| EXTERNAL_USERID | VARCHAR2(1024) | | External user name, in the case of network or external authentication |
| GLOBAL_USERID | VARCHAR2(32) | | Global user identifier for the user, for a user logged in as an enterprise user |
| CLIENT_PROGRAM_NAME | VARCHAR2(48) | | Name of the program used for the database session |
| DBLINK_INFO | VARCHAR2(4000) | | Value of `SYS_CONTEXT('USERENV','DBLINK_INFO')`. Valid if the connection was via a database link. |
| XS_USER_NAME | VARCHAR2(128) | | Name of the Real Application Security user |
| XS_SESSIONID | RAW(33) | | Identifer of the Real Application Security session |
| ENTRY_ID | NUMBER | | Numeric ID for each audit trail entry in the session |
| STATEMENT_ID | NUMBER | | Numeric ID for each statement run (a statement may cause many actions) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EVENT_TIMESTAMP | TIMESTAMP(6) WITH LOCAL TIME ZONE | | Timestamp of the creation of the audit trail entry in UTC |
| ACTION_NAME | VARCHAR2(64) | | Name of the action executed by the user. The name should be read in conjunction with the AUDIT_TYPE to understand the real action. |
| RETURN_CODE | NUMBER | | Oracle error code generated by the action. Zero if the action succeeded |
| OS_PROCESS | VARCHAR2(16) | | Operating system process identifier of the Oracle database process |
| TRANSACTION_ID | RAW(8) | | Transaction identifier of the transaction in which the object is modified |
| SCN | NUMBER | | System change number (SCN) of the query at the time of the event |
| EXECUTION_ID | VARCHAR2(64) | | Execution context identifier for each action |
| OBJECT_SCHEMA | VARCHAR2(30) | | Schema name of object affected by the action |
| OBJECT_NAME | VARCHAR2(128) | | Name of the object affected by the action |
| SQL_TEXT | CLOB | | SQL associated with the event |
| SQL_BINDS | CLOB | | List of bind variables, if any, associated with SQL_TEXT |
| APPLICATION_CONTEXTS | VARCHAR2(4000) | | Semicolon-seperated list of Application Context Namespace, Attribute, Value information in (APPCTX_NSPACE,APPCTX_ATTRIBUTE=<value>) format |
| CLIENT_IDENTIFIER | VARCHAR2(64) | | Client identifier in each Oracle session |
| NEW_SCHEMA | VARCHAR2(30) | | The schema of the object named in the NEW_NAME column |
| NEW_NAME | VARCHAR2(128) | | New name of object after RENAME, or name of underlying object (for example, CREATE INDEX owner.obj_name ON new_owner.new_name) |
| OBJECT_EDITION | VARCHAR2(30) | | Name of the edition containing the audited object |
| SYSTEM_PRIVILEGE_USED | VARCHAR2(1024) | | Comma-separated list of system privileges used to execute the action |
| SYSTEM_PRIVILEGE | VARCHAR2(40) | | System privilege granted/revoked by a GRANT/REVOKE statement |
| AUDIT_OPTION | VARCHAR2(40) | | AUDIT/NOAUDIT SQL command |
| OBJECT_PRIVILEGES | VARCHAR2(19) | | Object privileges granted/revoked by a GRANT/REVOKE statement |
| ROLE | VARCHAR2(30) | | Roles granted or revoked or set by GRANT/REVOKE/SET ROLE command |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TARGET_USER | VARCHAR2(30) | | User on whom the GRANT/REVOKE/AUDIT/ NOAUDIT statement was executed |
| EXCLUDED_USER | VARCHAR2(30) | | User who was excluded when the AUDIT/ NOAUDIT statement was executed |
| EXCLUDED_SCHEMA | VARCHAR2(30) | | Displays the schema of the excluded objects |
| EXCLUDED_OBJECT | VARCHAR2(128) | | Displays object excluded from the action |
| ADDITIONAL_INFO | VARCHAR2(4000) | | Text comment on the audit trail entry, if any |
| UNIFIED_AUDIT_POLICIES | VARCHAR2(4000) | | Lists the audit policies that caused the current audit record. For example, if SELECT ON SCOTT.EMP was configured from policy SCOTT_EMP_POL, for the SELECT event this column will display SCOTT_EMP_POL.<br>If more than one policy was configured, the list of policies that caused the event to be recorded in the audit trail are displayed in a comma-separated list. |
| FGA_POLICY_NAME | VARCHAR2(30) | | Fine-grained auditing (FGA) policy name that generated this FGA audit record |
| XS_INACTIVITY_TIMEOUT | NUMBER | | Inactivity timeout of the Real Application Security session |
| XS_ENTITY_TYPE | VARCHAR2(32) | | Type of the Real Application Security entity. Possible values are USER, ROLE, ROLESET, SECURITYCLASS, ACL, DATASECURITY, and NSTEMPLATE. |
| XS_TARGET_PRINCIPAL_NAME | VARCHAR2(30) | | Target principal name in Real Application Security operations. Possible operations are set verifier, set password, add proxy, remove proxy, switch user, assign user, create session,and grant roles. |
| XS_PROXY_USER_NAME | VARCHAR2(30) | | Name of the Real Application Security proxy user |
| XS_DATASEC_POLICY_NAME | VARCHAR2(30) | | Name of the Real Application Security data security policy enabled or disabled |
| XS_SCHEMA_NAME | VARCHAR2(30) | | Name of the schema in enable, disable data security policy and global callback operation |
| XS_CALLBACK_EVENT_TYPE | VARCHAR2(32) | | Real Application Security global callback event type |
| XS_PACKAGE_NAME | VARCHAR2(30) | | Real Application Security callback package name for the global callback |
| XS_PROCEDURE_NAME | VARCHAR2(30) | | Real Application Security callback procedure name for the global callback |
| XS_ENABLED_ROLE | VARCHAR2(30) | | The role that is enabled |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| XS_COOKIE | VARCHAR2(1024) | | Real Application Security session cookie |
| XS_NS_NAME | VARCHAR2(30) | | Name of the Real Application Security session namespace |
| XS_NS_ATTRIBUTE | VARCHAR2(4000) | | Name of the Real Application Security session namespace attribute |
| XS_NS_ATTRIBUTE_OLD_VAL | VARCHAR2(4000) | | The old value of the Real Application Security session namespace attribute |
| XS_NS_ATTRIBUTE_NEW_VAL | VARCHAR2(4000) | | The new value of the Real Application Security session namespace |
| DV_ACTION_CODE | NUMBER | | Numeric action type code for Database Vault |
| DV_ACTION_NAME | VARCHAR2(30) | | Name of the action whose numeric code appears in the DV_ACTION_CODE column |
| DV_EXTENDED_ACTION_CODE | NUMBER | | Numeric action type code for Database Vault administration |
| DV_GRANTEE | VARCHAR2(30) | | Name of the user whose Database Vault authorization was modified |
| DV_RETURN_CODE | NUMBER | | Database Vault specific error code |
| DV_ACTION_OBJECT_NAME | VARCHAR2(128) | | The unique name of the Database Vault object that was modified |
| DV_RULE_SET_NAME | VARCHAR2(90) | | The unique name of the rule set that was executing and caused the audit event to trigger |
| DV_COMMENT | VARCHAR2(4000) | | Text comment on the audit trail entry, providing more information about the statement audited |
| DV_FACTOR_CONTEXT | VARCHAR2(4000) | | An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered |
| DV_OBJECT_STATUS | VARCHAR2(1) | | Indicates whether a particular Database Vault object is enabled or disabled. For example, if a Database Vault administrator enables or disables a realm, then this event will be audited and the DV_OBJECT_STATUS value will show the status of the realm after the event occurred. Possible values for this column are:<br>• Y - The object is enabled<br>• N - The object is disabled |
| OLS_POLICY_NAME | VARCHAR2(30) | | Name of the Oracle Label Security (OLS) policy for which this audit record is generated |
| OLS_GRANTEE | VARCHAR2(30) | | Name of the user whose OLS authorization was modified |
| OLS_MAX_READ_LABEL | VARCHAR2(4000) | | Maximum read label assigned to a user |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OLS_MAX_WRITE_LABEL | VARCHAR2(4000) | | Maximum write label assigned to a user |
| OLS_MIN_WRITE_LABEL | VARCHAR2(4000) | | Minimum write label assigned to a user |
| OLS_PRIVILEGES_GRANTED | VARCHAR2(30) | | OLS privileges assigned to a user or a trusted stored procedure |
| OLS_PROGRAM_UNIT_NAME | VARCHAR2(30) | | Name of the trusted stored procedure whose authorization was modified or was executed |
| OLS_PRIVILEGES_USED | VARCHAR2(128) | | OLS privileges used for an event |
| OLS_STRING_LABEL | VARCHAR2(4000) | | String representation of the OLS label |
| OLS_LABEL_COMPONENT_TYPE | VARCHAR2(12) | | Type of the OLS label component |
| OLS_LABEL_COMPONENT_NAME | VARCHAR2(30) | | Name of the OLS label component |
| OLS_PARENT_GROUP_NAME | VARCHAR2(30) | | Name of the parent of the OLS group |
| OLS_OLD_VALUE | VARCHAR2(4000) | | Old value for OLS ALTER events |
| OLS_NEW_VALUE | VARCHAR2(4000) | | New value for OLS ALTER events |
| RMAN_SESSION_RECID | NUMBER | | RMAN session identifier. Together with RMAN_SESSION_STAMP uniquely identifies an RMAN job (note that this is not same as user session ID; the value is a recid in controlfile that identifies RMAN job) |
| RMAN_SESSION_STAMP | NUMBER | | Timestamp for the session |
| RMAN_OPERATION | VARCHAR2(20) | | The RMAN operation executed by the job. One row will be added for each distinct operation within an RMAN session. For example, a backup job would contain BACKUP in the RMAN_OPERATION column. |
| RMAN_OBJECT_TYPE | VARCHAR2(20) | | Type of objects involved for backup or restore/recover or change/delete/crosscheck commands.<br>It contains one of the following values. If RMAN command does not satisfy one of them, then preference is given in order, from top to bottom of the list:<br>• DB FULL<br>• RECVR AREA<br>• DB INCR<br>• DATAFILE FULL<br>• DATAFILE INCR<br>• ARCHIVELOG |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • CONTROLFILE<br>• SPFILE |
| RMAN_DEVICE_TYPE | VARCHAR2(5) | | Device involved in the RMAN job. It may be DISK or SBT_TAPE or * (An * indicates that more than one location is involved).<br>For a backup job, it will be the output device type. For other commands (such as restore or crosscheck), it will be the input device type. |
| DP_TEXT_PARAMETERS1 | VARCHAR2(512) | | Parameters during a Data Pump operation that have a text/string value. This may contain the values for:<br>• ACCESS METHODS<br>• DATA OPTIONS<br>• DUMPER DIRECTORY<br>• JOB_TYPE<br>• JOB VERSION<br>• MASTER TABLE<br>• METADATA_JOB_MODE<br>• PARTITION OPTIONS<br>• REMOTE LINK<br>• TABLE EXISTS<br>For descriptions and more information about the settings that can appear for these Data Pump text parameters, see Table 6-2. |
| DP_BOOLEAN_PARAMETERS1 | VARCHAR2(512) | | Parameters during a Data Pump operation that have a boolean value. This may contain the values for:<br>• DATA_ONLY - Boolean value for whether or not the operation processed data only (as opposed to metadata only, or metadata and data combined)<br>• DUMPFILE_PRESENT - Denotes whether a dump file exists. Typically, it indicates whether a network export in which no dumpfile is required.<br>• JOB_RESTARTED - Boolean that indicates if the export or import job had to be restarted<br>• MASTER_ONLY - Indicates whether the import job imported just the master table and then stopped the job so that the contents of the master table can be examined<br>• METADATA_ONLY - Boolean value for whether or not the operation processed metadata only (as opposed to data only, or metadata and data combined) |
| DIRECT_PATH_NUM_COLUMNS_LOADED | NUMBER | | Shows the number of columns that were loaded using the SQL*Loader direct path load method |

***Table 6-2 Data Pump Text Parameter Descriptions***

| Parameter | Description |
|---|---|
| ACCESS METHOD | The method used to load the data. Settings can be:<br>• AUTOMATIC: Enables Oracle Data Pump to determine the optimal load method<br>• DIRECT_PATH : Uses the direct path API to pass the data to be loaded.<br>• EXTERNAL_TABLE: Loads data using the external tables option.<br>• CONVENTIONAL: Loads the data using SQL INSERT statements |
| DATA OPTIONS | Indicates how certain types of data were handled during import operations. Settings are in bit-mask format, which are as follows:<br>• 1 (SKIP_CONSTRAINT_ERRORS): Specifies that the import operation proceeded even if non-deferred constraint violations were encountered.<br>• 2 (XML_CLOBS): Allows for the export of the XML type data as a CLOB. This setting is deprecated in Oracle Database 12*c* Release 1 (12.1). See *Oracle Database Utilities* for more information.<br>• 8 (DISABLE_APPEND_HINT): The import operation did not use the APPEND hint while loading a data object.<br>• 16 (REJECT_ROWS_WITH_REPL_CHAR): Warnings are issued when the replacement character may be used and an option was added to reject data rows where the replacement character was used during a Data Pump import. This situation can occur if different character sets are used for the export/import process |
| DUMPER DIRECTORY | Not in use |
| JOB_TYPE | Is either EXPORT or IMPORT |
| JOB VERSION | Specifies the version of database objects that were imported |
| MASTER TABLE | Indicates the name of the master table. By default, it appears as follows for export operations:<br>*schema_name*.SYS_EXPORT_TABLE_*n*<br>For import operations, it appears as follows:<br>*schema_name*.SYS_IMPORT_TABLE_*n*<br>The *n* represents a numeric value of 01. If 01 is in use, the number is incremented with 02, 03, and so on. |
| METADATA_JOB_MODE | Type of export or import operation. For example a table export would be TABLE_EXPORT |
| PARTITION OPTIONS | Indicates how table partitions were created during an import operation. Settings can be:<br>• NONE: The tables were created as they existed on the system from which the export operation was performed.<br>• DEPARTITION: Each partition or subpartition was promoted to a new individual table. |

| Parameter | Description |
|---|---|
| | • MERGE: All partitions and subpartitions were merged into one table |
| REMOTE LINK | Indicates that the export was performed from a (source) database identified by a valid database link. The data from the source database instance was written to a dump file set on the connected database instance. |
| TABLE EXISTS | Indicates the action that was taken on an import operation when the target table already existed. The values are as follows:<br>• REPLACE<br>• TRUNCATE<br>• SKIP<br>• APPEND |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_AUDIT_MGMT package

# USABLE_EDITIONS

USABLE_EDITIONS describes the usable editions of the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| EDITION_NAME | VARCHAR2(128) | NOT NULL | Name of the edition |
| PARENT_EDITION_NAME | VARCHAR2(128) | | Name of the parent edition for this edition |

**See Also:**

*Oracle Database Development Guide* for more information about editions

# USER_ADDM_FDG_BREAKDOWN

USER_ADDM_FDG_BREAKDOWN describes the contribution for each finding from the different instances owned by the current user. Its columns are the same as those in "DBA_ADDM_FDG_BREAKDOWN".

# USER_ADDM_FINDINGS

USER_ADDM_FINDINGS displays the ADDM findings discovered by the advisors owned by the current user. Each row for ADDM tasks in the related USER_ADVISOR_FINDINGS view has a corresponding row in this view. Its columns (except for OWNER) are the same as those in "DBA_ADDM_FINDINGS".

## USER_ADDM_INSTANCES

USER_ADDM_INSTANCES provides instance-level information for ADDM tasks that finished executing in all instances owned by the current user. Its columns are the same as those in "DBA_ADDM_INSTANCES".

## USER_ADDM_TASK_DIRECTIVES

USER_ADDM_TASK_DIRECTIVES displays information about ADDM task directives owned by the current user. Its columns (except for USERNAME and SEQ_ID) are the same as those in DBA_ADDM_TASK_DIRECTIVES.

**See Also:**

"DBA_ADDM_TASK_DIRECTIVES"

## USER_ADDM_TASKS

USER_ADDM_TASKS displays information about the ADDM tasks owned by the current user. The view contains one row for each row in the related USER_ADVISOR_TASKS view that has ADVISOR_NAME=ADDM and STATUS=COMPLETED. Its columns (except for OWNER) are the same as those in "DBA_ADVISOR_TASKS".

## USER_ADVISOR_ACTIONS

USER_ADVISOR_ACTIONS displays information about the actions associated with the recommendations owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_ACTIONS.

**See Also:**

"DBA_ADVISOR_ACTIONS"

## USER_ADVISOR_DIR_TASK_INST

USER_ADVISOR_DIR_TASK_INST displays information about all task directive instances owned by the current user. Its columns (except for SEQ_ID and USERNAME) are the same as those in "DBA_ADVISOR_DIR_TASK_INST".

## USER_ADVISOR_EXEC_PARAMETERS

USER_ADVISOR_EXEC_PARAMETERS displays the parameter values used for past executions of tasks owned by the current user. Its columns (except for OWNER) are the same as those in "DBA_ADVISOR_EXEC_PARAMETERS".

## USER_ADVISOR_EXECUTIONS

USER_ADVISOR_EXECUTIONS displays metadata information for tasks owned by the current user. Its columns (except for OWNER) are the same as those in "DBA_ADVISOR_EXECUTIONS".

## USER_ADVISOR_FDG_BREAKDOWN

USER_ADVISOR_FDG_BREAKDOWN describes the contribution from the different instances to the findings for each ADDM task owned by the current user. Its columns are the same as those in DBA_ADVISOR_FDG_BREAKDOWN.

**See Also:**

"DBA_ADVISOR_FDG_BREAKDOWN"

## USER_ADVISOR_FINDINGS

USER_ADVISOR_FINDINGS displays the findings discovered by the advisors owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_FINDINGS.

**See Also:**

"DBA_ADVISOR_FINDINGS"

## USER_ADVISOR_JOURNAL

USER_ADVISOR_JOURNAL displays the journal entries for the tasks owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_JOURNAL.

**See Also:**

"DBA_ADVISOR_JOURNAL"

## USER_ADVISOR_LOG

USER_ADVISOR_LOG displays information about the current state of the tasks owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_LOG.

**See Also:**

"DBA_ADVISOR_LOG"

## USER_ADVISOR_OBJECTS

USER_ADVISOR_OBJECTS displays information about the objects currently referenced by the advisors owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_OBJECTS.

**See Also:**

"DBA_ADVISOR_OBJECTS"

## USER_ADVISOR_PARAMETERS

USER_ADVISOR_PARAMETERS displays the task parameters and their current values for the tasks owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_PARAMETERS.

**See Also:**

"DBA_ADVISOR_PARAMETERS"

## USER_ADVISOR_RATIONALE

USER_ADVISOR_RATIONALE displays information about the rationales for the recommendations owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_RATIONALE.

**See Also:**

"DBA_ADVISOR_RATIONALE"

## USER_ADVISOR_RECOMMENDATIONS

USER_ADVISOR_RECOMMENDATIONS displays the results of an analysis of the recommendations owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_RECOMMENDATIONS.

**See Also:**

"DBA_ADVISOR_RECOMMENDATIONS"

## USER_ADVISOR_SQLA_REC_SUM

USER_ADVISOR_SQLA_REC_SUM displays recommendation rollup information for the workload objects owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLA_REC_SUM.

**See Also:**

"DBA_ADVISOR_SQLA_REC_SUM"

## USER_ADVISOR_SQLA_TABLES

USER_ADVISOR_SQLA_TABLES displays cross references between the workload statements and the tables referenced in the statement for the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLA_TABLES.

**See Also:**

"DBA_ADVISOR_SQLA_TABLES"

## USER_ADVISOR_SQLA_WK_MAP

USER_ADVISOR_SQLA_WK_MAP displays the workload references for the tasks owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLA_WK_MAP.

**See Also:**

"DBA_ADVISOR_SQLA_WK_MAP"

## USER_ADVISOR_SQLA_WK_STMTS

USER_ADVISOR_SQLA_WK_STMTS displays information about the workload objects owned by the current user after an Access Advisor analysis operation. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLA_WK_STMTS.

**See Also:**

"DBA_ADVISOR_SQLA_WK_STMTS"

## USER_ADVISOR_SQLPLANS

USER_ADVISOR_SQLPLANS displays the different SQL execution plans owned by the current user generated as part of an advisor analysis. Its columns are the same as those in "DBA_ADVISOR_SQLPLANS".

## USER_ADVISOR_SQLSTATS

USER_ADVISOR_SQLSTATS displays execution statistics owned by the current user for the test-execution of different SQL plans during the advisor analysis. Its columns are the same as those in "DBA_ADVISOR_SQLSTATS".

## USER_ADVISOR_SQLW_JOURNAL

USER_ADVISOR_SQLW_JOURNAL displays the journal entries for the workload objects owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLW_JOURNAL.

**See Also:**

"DBA_ADVISOR_SQLW_JOURNAL"

## USER_ADVISOR_SQLW_PARAMETERS

USER_ADVISOR_SQLW_PARAMETERS displays the workload parameters and their current values owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLW_PARAMETERS.

**See Also:**

"DBA_ADVISOR_SQLW_PARAMETERS"

## USER_ADVISOR_SQLW_STMTS

USER_ADVISOR_SQLW_STMTS displays rows that correspond to the statements in the workload owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLW_STMTS.

**See Also:**

"DBA_ADVISOR_SQLW_STMTS"

## USER_ADVISOR_SQLW_SUM

USER_ADVISOR_SQLW_SUM displays an aggregated picture of the SQLWkld workload objects owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLW_SUM.

**See Also:**

"DBA_ADVISOR_SQLW_SUM"

## USER_ADVISOR_SQLW_TABLES

USER_ADVISOR_SQLW_TABLES displays cross references between the workload statements and the tables referenced in the statement. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLW_TABLES.

**See Also:**

"DBA_ADVISOR_SQLW_TABLES"

## USER_ADVISOR_SQLW_TEMPLATES

USER_ADVISOR_SQLW_TEMPLATES displays an aggregated picture of the SQLWkld template objects owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_SQLW_TEMPLATES.

**See Also:**

"DBA_ADVISOR_SQLW_TEMPLATES"

## USER_ADVISOR_TASKS

USER_ADVISOR_TASKS displays information about the tasks owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_TASKS.

**See Also:**

"DBA_ADVISOR_TASKS"

## USER_ADVISOR_TEMPLATES

USER_ADVISOR_TEMPLATES displays information about the templates owned by the current user. Its columns (except for OWNER) are the same as those in DBA_ADVISOR_TEMPLATES.

**See Also:**

"DBA_ADVISOR_TEMPLATES"

## USER_ALL_TABLES

USER_ALL_TABLES describes the object tables and relational tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_ALL_TABLES.

**See Also:**

"ALL_ALL_TABLES"

## USER_APPLY_ERROR

USER_APPLY_ERROR displays information about the error transactions generated by apply processes visible to the current user. Its columns (except for SOURCE_ROOT_NAME) are the same as those in ALL_APPLY_ERROR.

**See Also:**

"ALL_APPLY_ERROR"

## USER_AQ_AGENT_PRIVS

USER_AQ_AGENT_PRIVS displays information about the registered AQ agents that are mapped to the current user. Its columns (except for DB_USERNAME) are the same as those in DBA_AQ_AGENT_PRIVS.

**See Also:**

"DBA_AQ_AGENT_PRIVS"

## USER_ARGUMENTS

USER_ARGUMENTS lists the arguments of the functions and procedures that are owned by the current user. Its columns (except for OWNER) are the same as those in ALL_ARGUMENTS.

**See Also:**

- "ALL_ARGUMENTS"
- "USER_PROCEDURES" for information about the functions and procedures that are owned by the current user

## USER_ASSEMBLIES

USER_ASSEMBLIES provides information about all assemblies owned by the current user. Its columns (except for OWNER) are the same as those in ALL_ASSEMBLIES.

**See Also:**

"ALL_ASSEMBLIES"

## USER_ASSOCIATIONS

USER_ASSOCIATIONS describes user-defined statistics associated with objects owned by the current user. Its columns are the same as those in ALL_ASSOCIATIONS.

**See Also:**

"ALL_ASSOCIATIONS"

# USER_ATTRIBUTE_TRANSFORMATIONS

USER_ATTRIBUTE_TRANSFORMATIONS displays information about the transformation functions for the transformations owned by the current user. Its columns (except for OWNER) are the same as those in ALL_ATTRIBUTE_TRANSFORMATIONS.

**See Also:**

"ALL_ATTRIBUTE_TRANSFORMATIONS"

# USER_AUDIT_OBJECT

USER_AUDIT_OBJECT displays audit trail records for the objects accessible to the current user. Its columns are the same as those in "DBA_AUDIT_OBJECT".

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

# USER_AUDIT_POLICIES

USER_AUDIT_POLICIES describes the fine-grained auditing policies on the tables and views owned by the current user. Its columns (except for OBJECT_SCHEMA) are the same as those in ALL_AUDIT_POLICIES.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

**See Also:**

"ALL_AUDIT_POLICIES"

## USER_AUDIT_POLICY_COLUMNS

USER_AUDIT_POLICY_COLUMNS describes the fine-grained auditing policy columns on the tables and views owned by the current user. Its columns are the same as those in ALL_AUDIT_POLICY_COLUMNS.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

**See Also:**

"ALL_AUDIT_POLICY_COLUMNS"

## USER_AUDIT_SESSION

USER_AUDIT_SESSION displays the audit trail records concerning connections and disconnections of the current user. Its columns are the same as those in "DBA_AUDIT_SESSION".

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

## USER_AUDIT_STATEMENT

USER_AUDIT_STATEMENT displays audit trail entries for the GRANT, REVOKE, AUDIT, NOAUDIT, and ALTER SYSTEM statements issued by the current user. Its columns are the same as those in "DBA_AUDIT_STATEMENT".

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

## USER_AUDIT_TRAIL

`USER_AUDIT_TRAIL` displays the standard audit trail entries related to the current user. Its columns are the same as those in "DBA_AUDIT_TRAIL".

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

## USER_AW_PS

`USER_AW_PS` describes the page spaces in the analytic workspaces owned by the current user. Its columns (except for `OWNER`) are the same as those in `ALL_AW_PS`.

**See Also:**

"ALL_AW_PS"

## USER_AWS

`USER_AWS` describes the analytic workspaces owned by the current user. Its columns (except for `OWNER`) are the same as those in `ALL_AWS`.

**See Also:**

"ALL_AWS"

## USER_BASE_TABLE_MVIEWS

USER_BASE_TABLE_MVIEWS describes the materialized views using materialized view logs owned by the current user. Its columns are the same as those in ALL_BASE_TABLE_MVIEWS.

**See Also:**

"ALL_BASE_TABLE_MVIEWS"

## USER_CATALOG

USER_CATALOG lists tables, views, clusters, synonyms, and sequences owned by the current user. Its columns are the same as those in "ALL_CATALOG".

## USER_CHANGE_NOTIFICATION_REGS

USER_CHANGE_NOTIFICATION_REGS describes the change notification registrations owned by the current user. Its columns (except for USERNAME) are the same as those in DBA_CHANGE_NOTIFICATION_REGS.

**See Also:**

"DBA_CHANGE_NOTIFICATION_REGS"

## USER_CLU_COLUMNS

USER_CLU_COLUMNS maps columns in the current user's tables to cluster columns. Its columns are the same as those in "DBA_CLU_COLUMNS".

## USER_CLUSTER_HASH_EXPRESSIONS

USER_CLUSTER_HASH_EXPRESSIONS lists hash functions for the hash clusters owned by the current user. Its columns are the same as those in "ALL_CLUSTER_HASH_EXPRESSIONS".

## USER_CLUSTERING_DIMENSIONS

USER_CLUSTERING_DIMENSIONS describes dimension tables associated with tables with an attribute clustering clause owned by the user. Its columns (except for OWNER) are the same as those in ALL_CLUSTERING_DIMENSIONS.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_DIMENSIONS"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## USER_CLUSTERING_JOINS

`USER_CLUSTERING_JOINS` describes joins to the dimension tables associated with tables with an attribute clustering clause owned by the user. Its columns (except for `OWNER`) are the same as those in `ALL_CLUSTERING_JOINS`.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_JOINS"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## USER_CLUSTERING_KEYS

`USER_CLUSTERING_KEYS` describes clustering keys for tables with an attribute clustering clause owned by the user. Its columns are the same as those in `ALL_CLUSTERING_KEYS`.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_KEYS"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## USER_CLUSTERING_TABLES

`USER_CLUSTERING_TABLES` describes the tables with an attribute clustering clause owned by the user. Its columns are the same as those in `ALL_CLUSTERING_TABLES`.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_CLUSTERING_TABLES"
- *Oracle Database Data Warehousing Guide* for information about attribute clustering with zone maps

## USER_CLUSTERS

`USER_CLUSTERS` describes all the clusters owned by the current user. Its columns are the same as those in "ALL_CLUSTERS".

## USER_CODE_ROLE_PRIVS

`USER_CODE_ROLE_PRIVS` describes all the roles that are associated with program units owned by current user. Its columns (except for `OWNER`) are the same as those in `ALL_CODE_ROLE_PRIVS`.

**See Also:**

"ALL_CODE_ROLE_PRIVS"

## USER_COL_COMMENTS

`USER_COL_COMMENTS` displays comments on the columns of the tables and views owned by the current user. Its columns (except for `OWNER`) are the same as those in `ALL_COL_COMMENTS`.

**See Also:**

"ALL_COL_COMMENTS"

## USER_COL_PENDING_STATS

`USER_COL_PENDING_STATS` describes the pending statistics of the columns owned by the current user. Its columns (except for `OWNER`) are the same as those in `ALL_COL_PENDING_STATS`.

**See Also:**

"ALL_COL_PENDING_STATS"

## USER_COL_PRIVS

USER_COL_PRIVS describes the column object grants for which the current user is the object owner, grantor, or grantee. Its columns are the same as those in DBA_COL_PRIVS.

**See Also:**

"DBA_COL_PRIVS"

## USER_COL_PRIVS_MADE

USER_COL_PRIVS_MADE describes the column object grants for which the current user is the object owner. Its columns (except for OWNER) are the same as those in ALL_COL_PRIVS_MADE.

**See Also:**

"ALL_COL_PRIVS_MADE"

## USER_COL_PRIVS_RECD

USER_COL_PRIVS_RECD describes the column object grants for which the current user is the grantee. Its columns (except for GRANTEE) are the same as those in ALL_COL_PRIVS_RECD.

**See Also:**

"ALL_COL_PRIVS_RECD"

## USER_COLL_TYPES

USER_COLL_TYPES describes named collection types (VARRAYs, nested tables, object tables, and so on) in the current user's schema. Its columns are the same as those in "ALL_COLL_TYPES".

## USER_COMPARISON

USER_COMPARISON displays information about the comparison objects owned by the current user. Its columns (except for OWNER) are the same as those in "DBA_COMPARISON".

## USER_COMPARISON_COLUMNS

USER_COMPARISON_COLUMNS displays information about the columns for the comparison objects owned by the current user. Its columns (except for OWNER) are the same as those in "DBA_COMPARISON_COLUMNS".

## USER_COMPARISON_ROW_DIF

USER_COMPARISON_ROW_DIF displays information about the differing rows in the comparison scans owned by the current user. Its columns are the same as those in "DBA_COMPARISON_ROW_DIF".

## USER_COMPARISON_SCAN

USER_COMPARISON_SCAN displays information about the comparison scans owned by the current user. Its columns (except for OWNER) are the same as those in "DBA_COMPARISON_SCAN".

## USER_COMPARISON_SCAN_VALUES

USER_COMPARISON_SCAN_VALUES displays information about the values for the comparison scans owned by the current user. Its columns (except for OWNER) are the same as those in "DBA_COMPARISON_SCAN_VALUES".

## USER_CONS_COLUMNS

USER_CONS_COLUMNS describes columns that are owned by the current user and that are specified in constraint definitions. Its columns are the same as those in "ALL_CONS_COLUMNS".

## USER_CONS_OBJ_COLUMNS

USER_CONS_OBJ_COLUMNS displays information about the types that object columns (or attributes) or collection elements have been constrained to, in the tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CONS_OBJ_COLUMNS.

**See Also:**

"ALL_CONS_OBJ_COLUMNS"

## USER_CONSTRAINTS

USER_CONSTRAINTS describes all constraint definitions on tables owned by the current user. Its columns are the same as those in "ALL_CONSTRAINTS".

## USER_CQ_NOTIFICATION_QUERIES

USER_CQ_NOTIFICATION_QUERIES describes the registered queries for the CQ notifications owned by the current user. Its columns (except for USERNAME) are the same as those in DBA_CQ_NOTIFICATION_QUERIES.

**See Also:**

"DBA_CQ_NOTIFICATION_QUERIES"

## USER_CREDENTIALS

USER_CREDENTIALS displays credentials owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CREDENTIALS.

**See Also:**

"ALL_CREDENTIALS"

## USER_CUBE_ATTR_VISIBILITY

USER_CUBE_ATTR_VISIBILITY describes the OLAP attributes visible for the dimensions, hierarchies, and levels owned by the current user Its columns (except for OWNER) are the same as those in ALL_CUBE_ATTR_VISIBILITY.

**See Also:**

"ALL_CUBE_ATTR_VISIBILITY"

## USER_CUBE_ATTRIBUTES

USER_CUBE_ATTRIBUTES describes the attributes for the OLAP cube dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_ATTRIBUTES.

**See Also:**

"ALL_CUBE_ATTRIBUTES"

## USER_CUBE_BUILD_PROCESSES

USER_CUBE_BUILD_PROCESSES describes the OLAP build processes and maintenance scripts owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_BUILD_PROCESSES.

**See Also:**

"ALL_CUBE_BUILD_PROCESSES"

## USER_CUBE_CALCULATED_MEMBERS

USER_CUBE_CALCULATED_MEMBERS describes the calculated members for the OLAP cube dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_CALCULATED_MEMBERS.

**See Also:**

"ALL_CUBE_CALCULATED_MEMBERS"

## USER_CUBE_DIM_LEVELS

USER_CUBE_DIM_LEVELS describes the OLAP dimension levels owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_DIM_LEVELS.

**See Also:**

"ALL_CUBE_DIM_LEVELS"

## USER_CUBE_DIM_MODELS

USER_CUBE_DIM_MODELS describes the models for the OLAP dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_DIM_MODELS.

**See Also:**

"ALL_CUBE_DIM_MODELS"

## USER_CUBE_DIM_VIEW_COLUMNS

USER_CUBE_DIM_VIEW_COLUMNS describes the columns of the relational views of the OLAP cube dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_DIM_VIEW_COLUMNS.

**See Also:**

"ALL_CUBE_DIM_VIEW_COLUMNS"

## USER_CUBE_DIM_VIEWS

USER_CUBE_DIM_VIEWS describes the relational views of the OLAP dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_DIM_VIEWS.

**See Also:**

"ALL_CUBE_DIM_VIEWS"

## USER_CUBE_DIMENSIONALITY

USER_CUBE_DIMENSIONALITY describes the dimension order for the OLAP cubes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_DIMENSIONALITY.

**See Also:**

"ALL_CUBE_DIMENSIONALITY"

## USER_CUBE_DIMENSIONS

USER_CUBE_DIMENSIONS describes the OLAP cube dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_DIMENSIONS.

**See Also:**

"ALL_CUBE_DIMENSIONS"

## USER_CUBE_HIER_LEVELS

USER_CUBE_HIER_LEVELS describes the hierarchy levels for the OLAP cube dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_HIER_LEVELS.

**See Also:**

"ALL_CUBE_HIER_LEVELS"

## USER_CUBE_HIER_VIEW_COLUMNS

USER_CUBE_HIER_VIEW_COLUMNS describes the columns of the relational hierarchy views of the OLAP cube dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_HIER_VIEW_COLUMNS.

**See Also:**

"ALL_CUBE_HIER_VIEW_COLUMNS"

## USER_CUBE_HIER_VIEWS

USER_CUBE_HIER_VIEWS describes the hierarchies for the OLAP cube dimensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_HIER_VIEWS.

**See Also:**

"ALL_CUBE_HIER_VIEWS"

## USER_CUBE_HIERARCHIES

USER_CUBE_HIERARCHIES describes the OLAP dimension hierarchies owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_HIERARCHIES.

**See Also:**

"ALL_CUBE_HIERARCHIES"

## USER_CUBE_MEASURES

USER_CUBE_MEASURES describes the measures for the OLAP cubes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_MEASURES.

**See Also:**

"ALL_CUBE_MEASURES"

## USER_CUBE_NAMED_BUILD_SPECS

USER_CUBE_NAMED_BUILD_SPECS describes the OLAP cube named build specifications in the database that are owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_NAMED_BUILD_SPECS.

**See Also:**

"ALL_CUBE_NAMED_BUILD_SPECS"

## USER_CUBE_SUB_PARTITION_LEVELS

USER_CUBE_SUB_PARTITION_LEVELS describes the OLAP secondary partition levels in the database that are owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_SUB_PARTITION_LEVELS.

**See Also:**

"ALL_CUBE_SUB_PARTITION_LEVELS"

## USER_CUBE_VIEW_COLUMNS

USER_CUBE_VIEW_COLUMNS describes the columns of relational views of OLAP cubes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_VIEW_COLUMNS.

**See Also:**

"ALL_CUBE_VIEW_COLUMNS"

## USER_CUBE_VIEWS

USER_CUBE_VIEWS describes the relational views of the OLAP cubes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBE_VIEWS.

**See Also:**

"ALL_CUBE_VIEWS"

## USER_CUBES

USER_CUBES describes the OLAP cubes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_CUBES.

**See Also:**

"ALL_CUBES"

## USER_DATAPUMP_JOBS

USER_DATAPUMP_JOBS displays the Data Pump jobs owned by the current user. Its columns (except for OWNER_NAME) are the same as those in DBA_DATAPUMP_JOBS.

**See Also:**

"DBA_DATAPUMP_JOBS"

## USER_DB_LINKS

USER_DB_LINKS describes the database links owned by the current user. Its columns are the same as those in ALL_DB_LINKS except that it does not have the OWNER column. It also displays an additional column, PASSWORD, which is no longer used and for which nothing is returned. The PASSWORD column is maintained for backward compatibility only.

**See Also:**

"ALL_DB_LINKS"

## USER_DBFS_HS

USER_DBFS_HS shows all Database File System (DBFS) hierarchical stores owned by the current user. Its columns (except for STOREOWNER) are the same as those in DBA_DBFS_HS.

**See Also:**

"DBA_DBFS_HS"

## USER_DBFS_HS_COMMANDS

USER_DBFS_HS_COMMANDS shows all the registered store commands for all Database File system (DBFS) hierarchical stores owned by current user. Its columns (except for STOREOWNER) are the same as those in DBA_DBFS_HS_COMMANDS.

**See Also:**

"DBA_DBFS_HS_COMMANDS"

## USER_DBFS_HS_FILES

USER_DBFS_HS_FILES displays files in the Database File System (DBFS) hierarchical store owned by the current user and their location on the back-end device.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PATH | VARCHAR2(1024) | | Path name of the file |
| SEQUENCENUMBER | NUMBER | | Sequence number of this piece of the file |
| STARTOFFSET | NUMBER | | Begin offset of this piece in the tarball |
| ENDOFFSET | NUMBER | | End offset of this piece in the tarball |
| TARBALLID | NUMBER | | Tarball ID |
| BACKUPFILENAME | VARCHAR2(256) | | File on back end in which this tarball is located |
| TARSTARTOFFSET | NUMBER | | Begin offset of this tarball in the backup file |
| TARENDOFFSET | NUMBER | | End offset of this tarball in the backup file |

## USER_DBFS_HS_FIXED_PROPERTIES

USER_DBFS_HS_FIXED_PROPERTIES shows non-modifiable properties of all Databaase File System (DBFS) hierarchical stores owned by current user. Its columns

(except for STORE_OWNER) are the same as those in DBA_DBFS_HS_FIXED_PROPERTIES.

**See Also:**

"DBA_DBFS_HS_FIXED_PROPERTIES"

## USER_DBFS_HS_PROPERTIES

USER_DBFS_HS_PROPERTIES shows modifiable properties of all Database File System (DBFS) hierarchical stores owned by current user. Its columns (except for STOREOWNER) are the same as those in DBA_DBFS_HS_PROPERTIES.

**See Also:**

"DBA_DBFS_HS_PROPERTIES"

## USER_DEPENDENCIES

USER_DEPENDENCIES describes dependencies between procedures, packages, functions, package bodies, and triggers owned by the current user, including dependencies on views created without any database links. Its columns are the same as those in "ALL_DEPENDENCIES".

## USER_DIM_ATTRIBUTES

USER_DIM_ATTRIBUTES describes the relationship between dimension levels and functionally dependent columns in the current user's schema. The level columns and the dependent column must be in the same table. This view's columns are the same as those in "ALL_DIM_ATTRIBUTES".

## USER_DIM_CHILD_OF

USER_DIM_CHILD_OF describes a hierarchical relationship of 1 to *n* between pairs of levels in dimensions owned by the current user. Its columns are the same as those in "ALL_DIM_CHILD_OF".

## USER_DIM_HIERARCHIES

USER_DIM_HIERARCHIES describes the dimension hierarchies owned by the current user. Its columns are the same as those in "ALL_DIM_HIERARCHIES".

## USER_DIM_JOIN_KEY

USER_DIM_JOIN_KEY describes the join between two dimension tables owned by the current user. The join is always specified between a parent dimension level column and a child column. This view's columns are the same as those in "ALL_DIM_JOIN_KEY".

## USER_DIM_LEVEL_KEY

USER_DIM_LEVEL_KEY describes columns of dimension levels owned by the current user. This view's columns are the same as those in "ALL_DIM_LEVEL_KEY".

## USER_DIM_LEVELS

USER_DIM_LEVELS describes the levels of dimensions owned by the current user. All columns of a dimension level must come from the same relation. This view's columns are the same as those in "ALL_DIM_LEVELS".

## USER_DIMENSIONS

USER_DIMENSIONS describes dimension objects in the user's schema. Its columns are the same as those in "ALL_DIMENSIONS".

## USER_EDITIONED_TYPES

USER_EDITIONED_TYPES lists the types that are editioned by default for the current user.. Its columns (except for SCHEMA) are the same as those in DBA_EDITIONED_TYPES.

**See Also:**

"DBA_EDITIONED_TYPES"

## USER_EDITIONING_VIEW_COLS

USER_EDITIONING_VIEW_COLS describes the relationship between the columns of the editioning views owned by the current user and the table columns to which they map. Its columns (except for OWNER) are the same as those in ALL_EDITIONING_VIEW_COLS.

**See Also:**

"ALL_EDITIONING_VIEW_COLS"

## USER_EDITIONING_VIEW_COLS_AE

USER_EDITIONING_VIEW_COLS_AE describes the relationship between the columns of the editioning views (across all editions) owned by the current user and the table columns to which they map. Its columns (except for OWNER) are the same as those in ALL_EDITIONING_VIEW_COLS_AE.

**See Also:**

"ALL_EDITIONING_VIEW_COLS_AE"

## USER_EDITIONING_VIEWS

USER_EDITIONING_VIEWS describes the editioning views owned by the current user. Its columns (except for OWNER) are the same as those in ALL_EDITIONING_VIEWS.

**See Also:**

"ALL_EDITIONING_VIEWS"

## USER_EDITIONING_VIEWS_AE

USER_EDITIONING_VIEWS_AE describes the editioning views (across all editions) owned by the current user. Its columns (except for OWNER) are the same as those in ALL_EDITIONING_VIEWS_AE.

**See Also:**

"ALL_EDITIONING_VIEWS_AE"

## USER_ENCRYPTED_COLUMNS

USER_ENCRYPTED_COLUMNS maintains encryption algorithm information for all encrypted columns in all tables in the user's schema. Its columns (except for OWNER) are the same as those in "ALL_ENCRYPTED_COLUMNS".

## USER_EPG_DAD_AUTHORIZATION

USER_EPG_DAD_AUTHORIZATION describes the DADs that are authorized to use the user's privileges. Its columns (except for USERNAME) are the same as those in "DBA_EPG_DAD_AUTHORIZATION".

## USER_ERROR_TRANSLATIONS

USER_ERROR_TRANSLATIONS describes all error translations owned by the user. Its columns (except for OWNER) are the same as those in ALL_ERROR_TRANSLATIONS.

**See Also:**

"ALL_ERROR_TRANSLATIONS"

## USER_ERRORS

USER_ERRORS describes the current errors on the stored objects owned by the current user. Its columns (except for OWNER) are the same as those in ALL_ERRORS.

**See Also:**

"ALL_ERRORS"

## USER_ERRORS_AE

USER_ERRORS_AE describes the current errors on the stored objects (across all editions) owned by the current user. Its columns (except for OWNER) are the same as those in ALL_ERRORS_AE.

**See Also:**

"ALL_ERRORS_AE"

## USER_EVALUATION_CONTEXT_TABLES

USER_EVALUATION_CONTEXT_TABLES describes the tables in the rule evaluation contexts owned by the current user. Its columns (except for EVALUATION_CONTEXT_OWNER) are the same as those in ALL_EVALUATION_CONTEXT_TABLES.

**See Also:**

"ALL_EVALUATION_CONTEXT_TABLES"

## USER_EVALUATION_CONTEXT_VARS

USER_EVALUATION_CONTEXT_VARS describes the variables in the rule evaluation contexts owned by the current user. Its columns (except for EVALUATION_CONTEXT_OWNER) are the same as those in ALL_EVALUATION_CONTEXT_VARS.

**See Also:**

"ALL_EVALUATION_CONTEXT_VARS"

## USER_EVALUATION_CONTEXTS

USER_EVALUATION_CONTEXTS describes the rule evaluation contexts owned by the current user. Its columns (except for EVALUATION_CONTEXT_OWNER) are the same as those in ALL_EVALUATION_CONTEXTS.

**See Also:**

"ALL_EVALUATION_CONTEXTS"

## USER_EXTENTS

USER_EXTENTS describes the extents comprising the segments owned by the current user's objects. Its columns (except for OWNER, FILE_ID, BLOCK_ID, and RELATIVE_FNO) are the same as those in DBA_EXTENTS.

**See Also:**

"DBA_EXTENTS"

## USER_EXTERNAL_LOCATIONS

USER_EXTERNAL_LOCATIONS describes the locations (data sources) of the external tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_EXTERNAL_LOCATIONS.

**See Also:**

"ALL_EXTERNAL_LOCATIONS"

## USER_EXTERNAL_TABLES

USER_EXTERNAL_TABLES describes the external tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_EXTERNAL_TABLES.

**See Also:**

"ALL_EXTERNAL_TABLES"

## USER_FILE_GROUP_EXPORT_INFO

USER_FILE_GROUP_EXPORT_INFO shows export-related information for all file groups that the current user can manage. Its columns (except for FILE_GROUP_OWNER) are the same as those in "ALL_FILE_GROUP_EXPORT_INFO".

## USER_FILE_GROUP_FILES

USER_FILE_GROUP_FILES shows the file set for each versioned group owned by the current user. Its columns (except for FILE_GROUP_OWNER) are the same as those in"ALL_FILE_GROUP_FILES".

## USER_FILE_GROUP_TABLES

USER_FILE_GROUP_TABLES shows information about tables owned by the current user that can be imported using the file set. Its columns (except for FILE_GROUP_OWNER) are the same as those in "ALL_FILE_GROUP_TABLES".

## USER_FILE_GROUP_TABLESPACES

USER_FILE_GROUP_TABLESPACES shows information about the transportable tablespaces present (partially or completely) in the file set owned by the current user (when the file set contains dump files). Its columns (except for FILE_GROUP_OWNER) are the same as those in "ALL_FILE_GROUP_TABLESPACES".

## USER_FILE_GROUP_VERSIONS

USER_FILE_GROUP_VERSIONS shows top-level version information for all file groups owned by the current user. Its columns (except for FILE_GROUP_OWNER) are the same as those in "ALL_FILE_GROUP_VERSIONS".

## USER_FILE_GROUPS

USER_FILE_GROUPS shows top-level metadata about file groups owned by the current user. Its columns (except for FILE_GROUP_OWNER) are the same as those in "ALL_FILE_GROUPS".

## USER_FLASHBACK_ARCHIVE

USER_FLASHBACK_ARCHIVE describes flashback data archives, which consist of multiple tablespaces and historic data from all transactions against tracked tables. The content of this view depends on the privileges of the user who queries it, as follows:

- If the user has the FLASHBACK ARCHIVE ADMINISTER system privilege, then USER_FLASHBACK_ARCHIVE describes the flashback archives for all users who have been granted the FLASHBACK ARCHIVE object privilege.
- If the user does not have the FLASHBACK ARCHIVE ADMINISTER system privilege, then USER_FLASHBACK_ARCHIVE describes flashback archives for which the current user has been granted the FLASHBACK ARCHIVE object privilege.

The columns of the USER_FLASHBACK_ARCHIVE view are the same as those in DBA_FLASHBACK_ARCHIVE.

**See Also:**

"DBA_FLASHBACK_ARCHIVE"

## USER_FLASHBACK_ARCHIVE_TABLES

USER_FLASHBACK_ARCHIVE_TABLES displays information about the tables owned by the current user that are enabled for Flashback Archive. Its columns are the same as those in DBA_FLASHBACK_ARCHIVE_TABLES.

**See Also:**

"DBA_FLASHBACK_ARCHIVE_TABLES"

## USER_FLASHBACK_TXN_REPORT

USER_FLASHBACK_TXN_REPORT displays information about the compensating transactions owned by the current user that have been committed in the database. Its columns (except for USERNAME) are the same as those in DBA_FLASHBACK_TXN_REPORT.

**See Also:**

"DBA_FLASHBACK_TXN_REPORT"

## USER_FLASHBACK_TXN_STATE

USER_FLASHBACK_TXN_STATE displays information about the compensating status of the transactions owned by the current user. Its columns (except for USERNAME) are the same as those in DBA_FLASHBACK_TXN_STATE.

**See Also:**

"DBA_FLASHBACK_TXN_STATE"

## USER_FREE_SPACE

USER_FREE_SPACE describes the free extents in the tablespaces accessible to the current user. Its columns are the same as those in DBA_FREE_SPACE.

**See Also:**

"DBA_FREE_SPACE"

## USER_GOLDENGATE_PRIVILEGES

USER_GOLDENGATE_PRIVILEGES displays details about Oracle GoldenGate privileges. Its columns (except for USERNAME) are the same as those in ALL_GOLDENGATE_PRIVILEGES.

**See Also:**

"ALL_GOLDENGATE_PRIVILEGES"

## USER_HEAT_MAP_SEG_HISTOGRAM

USER_HEAT_MAP_SEG_HISTOGRAM displays segment access information for segments owned by the user. Its columns (except for OWNER) are the same as those in ALL_HEAT_MAP_SEG_HISTOGRAM.

**See Also:**

"ALL_HEAT_MAP_SEG_HISTOGRAM"

## USER_HEAT_MAP_SEGMENT

USER_HEAT_MAP_SEGMENT displays the latest segment access time for all segments owned by the user. Its columns (except for OWNER) are the same as those in ALL_HEAT_MAP_SEGMENT.

**See Also:**

"ALL_HEAT_MAP_SEGMENT"

## USER_HISTOGRAMS

USER_HISTOGRAMS is a synonym for USER_TAB_HISTOGRAMS.

**See Also:**

"USER_TAB_HISTOGRAMS"

## USER_HOST_ACES

USER_HOST_ACES describes the status of access control entries for the current user to access network hosts through PL/SQL network utility packages. Its columns (except for ACE_ORDER, START_DATE, END_DATE, GRANT_TYPE, INVERTED_PRINCIPAL, PRINCIPAL, PRINCIPAL_TYPE, and STATUS) are the same as those in DBA_HOST_ACES.

**See Also:**

"DBA_HOST_ACES"

## USER_IDENTIFIERS

USER_IDENTIFIERS displays information about the identifiers in the stored objects owned by the current user. Its columns (except for OWNER) are the same as those in ALL_IDENTIFIERS.

**See Also:**

"ALL_IDENTIFIERS"

## USER_ILMDATAMOVEMENTPOLICIES

USER_ILMDATAMOVEMENTPOLICIES contains information specific to data movement-related attributes of an Automatic Data Optimization policy for a user. Its columns are the same as those in DBA_ILMDATAMOVEMENTPOLICIES.

**Note:**

Automatic Data Optimization is not supported with a CDB.

**See Also:**

"DBA_ILMDATAMOVEMENTPOLICIES"

# USER_ILMEVALUATIONDETAILS

USER_ILMEVALUATIONDETAILS displays details on evaluation of Automatic Data Optimization policies considered for Automatic Data Optimization tasks for a user. It also shows the job name that executes the policy, in case the policy was selected for execution. If the policy was not selected for execution, this view provides a reason. Its columns are the same as those in DBA_ILMEVALUATIONDETAILS.

**Note:**

Automatic Data Optimization is not supported with a CDB.

**See Also:**

"DBA_ILMEVALUATIONDETAILS"

# USER_ILMOBJECTS

USER_ILMOBJECTS displays all the Automatic Data Optimization policies and objects for a user. Many objects inherit policies via their parent objects or because they were created in a particular tablespace. This view provides a mapping between the policies and objects and indicates whether a policy is inherited by an object or is directly specified on it. Its columns are the same as those in DBA_ILMOBJECTS.

**Note:**

Automatic Data Optimization is not supported with a CDB.

**See Also:**

"DBA_ILMOBJECTS"

# USER_ILMPOLICIES

USER_ILMPOLICIES displays details about Automatic Data Optimization policies owned by the user. The view contains common details relevant to all types of Automatic Data Optimization policies, not just details relevant to the data movement-related Automatic Data Optimization policies. Its columns are the same as those in DBA_ILMPOLICIES.

**Note:**

Automatic Data Optimization is not supported with a CDB.

**See Also:**

"DBA_ILMPOLICIES"

## USER_ILMRESULTS

USER_ILMRESULTS displays information on data movement-related Automatic Data Optimization jobs for tasks created by the user. Its columns are the same as those in DBA_ILMRESULTS.

**Note:**

Automatic Data Optimization is not supported with a CDB.

**See Also:**

"DBA_ILMRESULTS"

## USER_ILMTASKS

USER_ILMTASKS displays information on Automatic Data Optimization tasks created by a user. Its columns are the same as those in DBA_ILMTASKS.

**Note:**

Automatic Data Optimization is not supported with a CDB.

**See Also:**

"DBA_ILMTASKS"

## USER_IND_COLUMNS

USER_IND_COLUMNS describes the columns of the indexes owned by the current user and columns of indexes on tables owned by the current user. Its columns (except for INDEX_OWNER and TABLE_OWNER) are the same as those in "ALL_IND_COLUMNS".

## USER_IND_EXPRESSIONS

USER_IND_EXPRESSIONS describes expressions of function-based indexes on tables owned by the current user. Its columns (except for INDEX_OWNER and TABLE_OWNER) are the same as those in "ALL_IND_EXPRESSIONS".

## USER_IND_PARTITIONS

USER_IND_PARTITIONS displays, for each index partition owned by the current user, the partition-level partitioning information, the storage parameters for the partition, and various partition statistics generated by the DBMS_STATS package. Its columns are the same as those in "ALL_IND_PARTITIONS".

## USER_IND_PENDING_STATS

USER_IND_PENDING_STATS describes pending statistics for all tables, partitions, and subpartitions owned by the current user and collected using the DBMS_STATS package. Its columns (except for OWNER) are the same as those in "ALL_IND_PENDING_STATS".

## USER_IND_STATISTICS

USER_IND_STATISTICS displays optimizer statistics for the indexes on the tables owned by the current user and collected using the DBMS_STATS package. Its columns (except for OWNER) are the same as those in ALL_IND_STATISTICS.

**See Also:**

"ALL_IND_STATISTICS"

## USER_IND_SUBPARTITIONS

USER_IND_SUBPARTITIONS displays, for each index subpartition owned by the current user, the subpartition-level partitioning information, the storage parameters for the subpartition, and various subpartition statistics generated by the DBMS_STATS package. Its columns are the same as those in "ALL_IND_SUBPARTITIONS".

## USER_INDEXES

USER_INDEXES describes indexes owned by the current user. To gather statistics for this view, use the DBMS_STATS package. This view supports parallel partitioned index scans. Its columns (except for OWNER) are the same as those in "ALL_INDEXES".

## USER_INDEXTYPE_ARRAYTYPES

USER_INDEXTYPE_ARRAYTYPES displays information about the array types specified by the indextypes owned by the current user. Its columns are the same as those in ALL_INDEXTYPE_ARRAYTYPES.

**See Also:**

"ALL_INDEXTYPE_ARRAYTYPES"

## USER_INDEXTYPE_COMMENTS

USER_INDEXTYPE_COMMENTS displays comments for the user-defined indextypes owned by the current user. Its columns are the same as those in ALL_INDEXTYPE_COMMENTS.

**See Also:**

"ALL_INDEXTYPE_COMMENTS"

## USER_INDEXTYPE_OPERATORS

USER_INDEXTYPE_OPERATORS lists all the operators supported by indextypes owned by the current user. Its columns are the same as those in ALL_INDEXTYPE_OPERATORS.

**See Also:**

"ALL_INDEXTYPE_OPERATORS"

## USER_INDEXTYPES

USER_INDEXTYPES describes the indextypes owned by the current user. Its columns are the same as those in ALL_INDEXTYPES.

**See Also:**

"ALL_INDEXTYPES"

## USER_INTERNAL_TRIGGERS

USER_INTERNAL_TRIGGERS describes the internal triggers on all tables owned by the current user. Its columns are the same as those in "ALL_INTERNAL_TRIGGERS".

## USER_JAVA_ARGUMENTS

USER_JAVA_ARGUMENTS displays argument information about the stored Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_ARGUMENTS.

**See Also:**

"ALL_JAVA_ARGUMENTS"

## USER_JAVA_CLASSES

USER_JAVA_CLASSES displays class level information about the stored Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_CLASSES.

**See Also:**

"ALL_JAVA_CLASSES"

# USER_JAVA_COMPILER_OPTIONS

USER_JAVA_COMPILER_OPTIONS displays information about the native compiler options owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_COMPILER_OPTIONS.

**See Also:**

"ALL_JAVA_COMPILER_OPTIONS"

# USER_JAVA_DERIVATIONS

USER_JAVA_DERIVATIONS displays mapping information about Java source objects and their derived Java class objects and Java resource objects for the Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_DERIVATIONS.

**See Also:**

"ALL_JAVA_DERIVATIONS"

# USER_JAVA_FIELDS

USER_JAVA_FIELDS displays field information about the stored Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_FIELDS.

**See Also:**

"ALL_JAVA_FIELDS"

# USER_JAVA_IMPLEMENTS

USER_JAVA_IMPLEMENTS describes interfaces implemented by the stored Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_IMPLEMENTS.

**See Also:**

"ALL_JAVA_IMPLEMENTS"

# USER_JAVA_INNERS

USER_JAVA_INNERS displays information about inner classes referred to by the stored Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_INNERS.

**See Also:**

"ALL_JAVA_INNERS"

## USER_JAVA_LAYOUTS

USER_JAVA_LAYOUTS displays class layout information about the stored Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_LAYOUTS.

**See Also:**

"ALL_JAVA_LAYOUTS"

## USER_JAVA_METHODS

USER_JAVA_METHODS displays method information about the stored Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_METHODS.

**See Also:**

"ALL_JAVA_METHODS"

## USER_JAVA_NCOMPS

USER_JAVA_NCOMPS displays ncomp-related information about the Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_NCOMPS.

**See Also:**

"ALL_JAVA_NCOMPS"

## USER_JAVA_POLICY

USER_JAVA_POLICY describes Java security permissions for the current user. Its columns are the same as those in DBA_JAVA_POLICY.

**See Also:**

"DBA_JAVA_POLICY"

## USER_JAVA_RESOLVERS

USER_JAVA_RESOLVERS displays information about resolvers of the Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_RESOLVERS.

**See Also:**

"ALL_JAVA_RESOLVERS"

## USER_JAVA_THROWS

USER_JAVA_THROWS displays information about exceptions thrown from methods of the Java classes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_JAVA_THROWS.

**See Also:**

"ALL_JAVA_THROWS"

## USER_JOBS

USER_JOBS describes the jobs owned by the current user. Its columns are the same as those in DBA_JOBS.

**See Also:**

"DBA_JOBS"

## USER_JOIN_IND_COLUMNS

USER_JOIN_IND_COLUMNS describes all join conditions owned by the current user. Its columns are the same as those in "ALL_JOIN_IND_COLUMNS" .

## USER_JSON_COLUMNS

USER_JSON_COLUMNS provides information on the JavaScript Object Notation (JSON) columns for which the user is the owner. Its columns (except for OWNER) are the same as those in ALL_JSON_COLUMNS. Each column owned by the user that has an IS JSON check constraint in an AND condition appears in this view. This view enables a user to find all the JSON columns that he or she owns.

For example, if a check constraint combines the IS JSON condition with another condition using logical condition OR, then the column is not listed in this view. In this case, it is not certain that the data in the column is JSON data. For example, the following constraint does not ensure that the data in column jcol is JSON data:

```
jcol is json OR length(jcol) < 1000
```

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "ALL_JSON_COLUMNS"
- *Oracle XML DB Developer's Guide* for more information about using JSON with Oracle Database

## USER_LIBRARIES

USER_LIBRARIES describes the libraries owned by the current user. Its columns (except for OWNER) are the same as those in ALL_LIBRARIES.

**See Also:**

"ALL_LIBRARIES"

## USER_LOB_PARTITIONS

USER_LOB_PARTITIONS displays the LOB partitions contained in the tables owned by the current user. Its columns are the same as those in "ALL_LOB_PARTITIONS".

## USER_LOB_SUBPARTITIONS

USER_LOB_SUBPARTITIONS displays partition-level attributes of the LOB data subpartitions owned by the current user. Its columns are the same as those in "ALL_LOB_SUBPARTITIONS".

- DBA_LOB_TEMPLATES describes all LOB subpartition templates in the database.

## USER_LOB_TEMPLATES

USER_LOB_TEMPLATES describes the LOB subpartition templates owned by the current user. Its columns (except for USER_NAME) are the same as those in ALL_LOB_TEMPLATES.

**See Also:**

"ALL_LOB_TEMPLATES"

## USER_LOBS

USER_LOBS displays the user's CLOBs and BLOBs contained in the user's tables. BFILEs are stored outside the database, so they are not described by this view. This view's columns are the same as those in "ALL_LOBS".

## USER_LOG_GROUP_COLUMNS

USER_LOG_GROUP_COLUMNS describes columns that are owned by the current user and that are specified in log groups. Its columns are the same as those in "ALL_LOG_GROUP_COLUMNS" .

## USER_LOG_GROUPS

USER_LOG_GROUPS describes log group definitions on tables owned by the current user. Its columns are the same as those in "ALL_LOG_GROUPS" .

## USER_MEASURE_FOLDER_CONTENTS

USER_MEASURE_FOLDER_CONTENTS describes the contents of the OLAP measure folders owned by the current user. Its columns (except for OWNER) are the same as those in ALL_MEASURE_FOLDER_CONTENTS.

**See Also:**

"ALL_MEASURE_FOLDER_CONTENTS"

## USER_MEASURE_FOLDER_SUBFOLDERS

USER_MEASURE_FOLDER_SUBFOLDERS describes the OLAP measure folders contained within the OLAP measure folders owned by the current user. Its columns (except for OWNER) are the same as those in ALL_MEASURE_FOLDER_SUBFOLDERS.

**See Also:**

"ALL_MEASURE_FOLDER_SUBFOLDERS"

## USER_MEASURE_FOLDERS

USER_MEASURE_FOLDERS describes the OLAP measure folders owned by the current user. Its columns (except for OWNER) are the same as those in ALL_MEASURE_FOLDERS.

**See Also:**

"ALL_MEASURE_FOLDERS"

## USER_METADATA_PROPERTIES

USER_METADATA_PROPERTIES describes OLAP metadata properties in the current user's schema. Its columns (except for OWNER) are the same as those in ALL_METADATA_PROPERTIES.

**See Also:**

"ALL_METADATA_PROPERTIES"

## USER_METHOD_PARAMS

USER_METHOD_PARAMS describes the method parameters of the object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_METHOD_PARAMS.

**See Also:**

"ALL_METHOD_PARAMS"

## USER_METHOD_RESULTS

USER_METHOD_RESULTS describes the method results of the object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_METHOD_RESULTS.

**See Also:**

"ALL_METHOD_RESULTS"

## USER_MINING_MODEL_ATTRIBUTES

USER_MINING_MODEL_ATTRIBUTES describes the mining model attributes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_MINING_MODEL_ATTRIBUTES.

**See Also:**

"ALL_MINING_MODEL_ATTRIBUTES"

## USER_MINING_MODEL_SETTINGS

USER_MINING_MODEL_SETTINGS describes the mining model settings owned by the current user. Its columns (except for OWNER) are the same as those in ALL_MINING_MODEL_SETTINGS.

**See Also:**

"ALL_MINING_MODEL_SETTINGS"

## USER_MINING_MODELS

USER_MINING_MODELS describes the mining models owned by the current user. Its columns (except for OWNER) are the same as those in ALL_MINING_MODELS.

**See Also:**

"ALL_MINING_MODELS"

## USER_MVIEW_AGGREGATES

USER_MVIEW_AGGREGATES describes the grouping functions (aggregated measures) that appear in the SELECT list of aggregated materialized views owned by the current user. Its columns are the same as those in "ALL_MVIEW_AGGREGATES".

## USER_MVIEW_ANALYSIS

USER_MVIEW_ANALYSIS describes all materialized views owned by the current user that potentially support query rewrite and that provide additional information for analysis by applications. Its columns are the same as those in "ALL_MVIEW_ANALYSIS".

**Note:**

This view excludes materialized views that reference remote tables or that include references to non-static values such as SYSDATE or USER. This view also excludes materialized views that were created as snapshots before Oracle8*i* and that were never altered to enable query rewrite.

## USER_MVIEW_COMMENTS

USER_MVIEW_COMMENTS displays comments on the materialized views owned by the current user. Its columns (except for OWNER) are the same as those in ALL_MVIEW_COMMENTS.

**See Also:**

"ALL_MVIEW_COMMENTS"

## USER_MVIEW_DETAIL_PARTITION

USER_MVIEW_DETAIL_PARTITION displays freshness information for all materialized views, with respect to a PCT detail partition, owned by the current user. Its columns are the same as those in "ALL_MVIEW_DETAIL_PARTITION".

## USER_MVIEW_DETAIL_RELATIONS

USER_MVIEW_DETAIL_RELATIONS represents the named detail relations that are either in the FROM list of a materialized view, or that are indirectly referenced through views in the FROM list. Its columns are the same as those in "ALL_MVIEW_DETAIL_RELATIONS".

## USER_MVIEW_DETAIL_SUBPARTITION

USER_MVIEW_DETAIL_SUBPARTITION displays freshness information for all materialized views, with respect to a PCT detail subpartition, owned by the current user. Its columns are the same as those in "ALL_MVIEW_DETAIL_SUBPARTITION".

## USER_MVIEW_JOINS

USER_MVIEW_JOINS describes a join between two columns in the WHERE clause of a subquery that defines a materialized view. Its columns are the same as those in "ALL_MVIEW_JOINS".

## USER_MVIEW_KEYS

USER_MVIEW_KEYS describes the columns or expressions in the SELECT list upon which materialized views in the current user's schema are based. Its columns are the same as those in "ALL_MVIEW_KEYS".

## USER_MVIEW_LOGS

USER_MVIEW_LOGS describes all materialized view logs owned by the current user. Its columns are the same as those in ALL_MVIEW_LOGS.

**See Also:**

"ALL_MVIEW_LOGS"

## USER_MVIEW_REFRESH_TIMES

USER_MVIEW_REFRESH_TIMES describes refresh times of the materialized views owned by the current user. Its columns are the same as those in ALL_MVIEW_REFRESH_TIMES.

**See Also:**

"ALL_MVIEW_REFRESH_TIMES"

## USER_MVIEWS

USER_MVIEWS describes all materialized views owned by the current user. Its columns are the same as those in ALL_MVIEWS.

**See Also:**

"ALL_MVIEWS"

## USER_NESTED_TABLE_COLS

USER_NESTED_TABLE_COLS describes the columns of the nested tables owned by the current user. Its columns (except for OWNER) are the same as those in

"ALL_NESTED_TABLE_COLS". To gather statistics for this view, use the DBMS_STATS package.

## USER_NESTED_TABLES

USER_NESTED_TABLES describes the nested tables in tables owned by the current user. Its columns are the same as those in "ALL_NESTED_TABLES".

## USER_NETWORK_ACL_PRIVILEGES

USER_NETWORK_ACL_PRIVILEGES describes the status of the network privileges for the current user to access network hosts.

**Note:**

This USER_NETWORK_ACL_PRIVILEGES view is deprecated in Oracle Database 12*c* Release 1 (12.1). Oracle recommends that you use the USER_HOST_ACES view, instead.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| HOST | VARCHAR2(1000) | NOT NULL | Network host |
| LOWER_PORT | NUMBER(5) | | Lower bound of the port range |
| UPPER_PORT | NUMBER(5) | | Upper bound of the port range |
| PRIVILEGE | CHAR(128) | | Network privilege |
| STATUS | VARCHAR2(7) | | Privilege status:<br>• DENIED<br>• GRANTED |

**See Also:**

"USER_HOST_ACES"

## USER_OBJ_AUDIT_OPTS

USER_OBJ_AUDIT_OPTS describes auditing options on all objects owned by the current user. Its columns (except for OWNER) are the same as those in "DBA_OBJ_AUDIT_OPTS".

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from `UNIFIED_AUDIT_TRAIL`.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

## USER_OBJ_COLATTRS

`USER_OBJ_COLATTRS` describes object columns and attributes contained in the tables owned by the current user. Its columns (except for `OWNER`) are the same as those in `ALL_OBJ_COLATTRS`.

**See Also:**

"ALL_OBJ_COLATTRS"

## USER_OBJECT_SIZE

`USER_OBJECT_SIZE` lists the sizes, in bytes, of various PL/SQL objects. Its columns are the same as those in "DBA_OBJECT_SIZE".

## USER_OBJECT_TABLES

`USER_OBJECT_TABLES` describes the object tables owned by the current user. Its columns (except for `OWNER`) are the same as those in `ALL_OBJECT_TABLES`.

**See Also:**

"ALL_OBJECT_TABLES"

## USER_OBJECT_USAGE

`USER_OBJECT_USAGE` displays statistics about index usage gathered from the database for the indexes owned by the current user. You can use this view to monitor index usage. All indexes owned by the current user that have been used at least once can be monitored and displayed in this view. Its columns (except for `OWNER`) are the same as those in `DBA_OBJECT_USAGE`.

**See Also:**

"DBA_OBJECT_USAGE"

## USER_OBJECTS

USER_OBJECTS describes all objects owned by the current user. Its columns (except for OWNER), are the same as those in ALL_OBJECTS.

**See Also:**

"ALL_OBJECTS"

## USER_OBJECTS_AE

USER_OBJECTS_AE describes the objects (across all editions) owned by the current user. Its columns (except for OWNER) are the same as those in ALL_OBJECTS_AE.

**See Also:**

"ALL_OBJECTS_AE"

## USER_OPANCILLARY

USER_OPANCILLARY provides ancillary information for operators owned by the current user. Its columns are the same as those in ALL_OPANCILLARY.

**See Also:**

"ALL_OPANCILLARY"

## USER_OPARGUMENTS

USER_OPARGUMENTS provides argument information for operator bindings owned by the current user. Its columns are the same as those in ALL_OPARGUMENTS.

**See Also:**

"ALL_OPARGUMENTS"

## USER_OPBINDINGS

USER_OPBINDINGS describes the binding functions and methods on the operators owned by the current user. Its columns are the same as those in ALL_OPBINDINGS.

**See Also:**

"ALL_OPBINDINGS"

## USER_OPERATOR_COMMENTS

USER_OPERATOR_COMMENTS displays comments for the user-defined operators owned by the current user. Its columns are the same as those in ALL_OPERATOR_COMMENTS.

**See Also:**

"ALL_OPERATOR_COMMENTS"

## USER_OPERATORS

USER_OPERATORS describes all operators owned by the current user. Its columns are the same as those in ALL_OPERATORS.

**See Also:**

"ALL_OPERATORS"

## USER_OUTLINE_HINTS

USER_OUTLINE_HINTS describes the set of hints stored in the outlines owned by the current user. Its columns (except for OWNER) are the same as those in DBA_OUTLINE_HINTS.

**See Also:**

"DBA_OUTLINE_HINTS"

## USER_OUTLINES

USER_OUTLINES describes the stored outlines owned by the current user. Its columns (except for OWNER) are the same as those in DBA_OUTLINES.

**See Also:**

"DBA_OUTLINES"

## USER_PARALLEL_EXECUTE_CHUNKS

USER_PARALLEL_EXECUTE_CHUNKS displays the chunks for tasks created by the current user. Its columns (except for TASK_OWNER) are the same as those in DBA_PARALLEL_EXECUTE_CHUNKS.

**See Also:**

"DBA_PARALLEL_EXECUTE_CHUNKS"

## USER_PARALLEL_EXECUTE_TASKS

USER_PARALLEL_EXECUTE_TASKS displays the tasks created by the current user. Its columns (except for TASK_OWNER) are the same as those in DBA_PARALLEL_EXECUTE_TASKS.

**See Also:**

"DBA_PARALLEL_EXECUTE_TASKS"

## USER_PART_COL_STATISTICS

USER_PART_COL_STATISTICS displays column statistics and histogram information for the table partitions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_PART_COL_STATISTICS.

**See Also:**

"ALL_PART_COL_STATISTICS"

## USER_PART_HISTOGRAMS

USER_PART_HISTOGRAMS displays the histogram data (endpoints per histogram) for the histograms on the table partitions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_PART_HISTOGRAMS.

**See Also:**

"ALL_PART_HISTOGRAMS"

## USER_PART_INDEXES

USER_PART_INDEXES displays the object-level partitioning information for the partitioned indexes owned by the current user. Its columns (except for OWNER) are the same as those in ALL_PART_INDEXES.

**See Also:**

"ALL_PART_INDEXES"

## USER_PART_KEY_COLUMNS

USER_PART_KEY_COLUMNS describes the partitioning key columns for the partitioned objects owned by the current user. Its columns (except for OWNER) are the same as those in ALL_PART_KEY_COLUMNS.

**See Also:**

"ALL_PART_KEY_COLUMNS"

## USER_PART_LOBS

USER_PART_LOBS displays table-level information about the partitioned LOBs owned by the current user, including default attributes for LOB data partitions. Its columns (except for TABLE_OWNER) are the same as those in ALL_PART_LOBS.

**See Also:**

"ALL_PART_LOBS"

## USER_PART_TABLES

USER_PART_TABLES displays the object-level partitioning information for the partitioned tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_PART_TABLES.

**See Also:**

"ALL_PART_TABLES"

## USER_PARTIAL_DROP_TABS

USER_PARTIAL_DROP_TABS describes all tables in the schema of the current user that have partially completed DROP COLUMN operations. Its columns are the same as those in "ALL_PARTIAL_DROP_TABS".

## USER_PASSWORD_LIMITS

USER_PASSWORD_LIMITS describes the password profile parameters that are assigned to the user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RESOURCE_NAME | VARCHAR2(32) | NOT NULL | Name of the password resource |
| LIMIT | VARCHAR2(40) | | Value of the resource limit |

## USER_PENDING_CONV_TABLES

USER_PENDING_CONV_TABLES describes the pending conversion tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_PENDING_CONV_TABLES.

**See Also:**

"ALL_PENDING_CONV_TABLES"

## USER_PLSQL_COLL_TYPES

USER_PLSQL_COLL_TYPES describes the user's own named PL/SQL collection types. Its columns (except for OWNER and CHAR_USED) are the same as those in ALL_PLSQL_COLL_TYPES.

**See Also:**

"ALL_PLSQL_COLL_TYPES"

## USER_PLSQL_OBJECT_SETTINGS

USER_PLSQL_OBJECT_SETTINGS displays information about the compiler settings for the stored objects owned by the current user. Its columns (except for OWNER) are the same as those in ALL_PLSQL_OBJECT_SETTINGS.

**See Also:**

"ALL_PLSQL_OBJECT_SETTINGS"

## USER_PLSQL_TYPE_ATTRS

USER_PLSQL_TYPE_ATTRS describes the attributes of the user's own PL/SQL types. Its columns (except for OWNER and CHAR_USED) are the same as those in ALL_PLSQL_TYPE_ATTRS.

**See Also:**

"ALL_PLSQL_TYPE_ATTRS"

## USER_PLSQL_TYPES

USER_PLSQL_TYPES describes the user's own PL/SQL types. Its columns (except for OWNER) are the same as those in ALL_PLSQL_TYPES.

**See Also:**

"ALL_PLSQL_TYPES"

## USER_POLICIES

USER_POLICIES describes all Oracle Virtual Private Database (VPD) security policies associated with objects owned by the current user. Its columns (except for OBJECT_OWNER) are the same as those in ALL_POLICIES.

**See Also:**

"ALL_POLICIES"

## USER_POLICY_ATTRIBUTES

USER_POLICY_ATTRIBUTES lists the attribute associations {Namespaces, Attributes} of all context-sensitive and shared-context sensitive Oracle Virtual Private Database (VPD) policies for synonyms, tables, or views owned by the user. Its columns are the same as those in ALL_POLICY_ATTRIBUTES.

**See Also:**

"ALL_POLICY_ATTRIBUTES"

## USER_POLICY_CONTEXTS

USER_POLICY_CONTEXTS describes the driving contexts defined for the synonyms, tables, and views owned by the current user. Its columns (except for OBJECT_OWNER) are the same as those in ALL_POLICY_CONTEXTS.

**See Also:**

"ALL_POLICY_CONTEXTS"

## USER_POLICY_GROUPS

USER_POLICY_GROUPS describes the policy groups defined for the synonyms, tables, and views owned by the current user. Its columns (except for OBJECT_OWNER) are the same as those in ALL_POLICY_GROUPS.

**See Also:**

"ALL_POLICY_GROUPS"

## USER_PRIVILEGE_MAP

USER_PRIVILEGE_MAP shows privilege (auditing option) type codes for object privileges that can be granted on a user. This table can be used to map privilege type numbers to type names.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| PRIVILEGE | NUMBER | NOT NULL | A numeric privilege (auditing option) type code |
| NAME | VARCHAR2(40) | NOT NULL | Name of the type of privilege (auditing option) |

## USER_PROCEDURES

USER_PROCEDURES lists all functions and procedures that are owned by the current user, along with their associated properties. Its columns (except OWNER) are the same as those in "ALL_PROCEDURES".

**See Also:**

"USER_ARGUMENTS" for information about the arguments of the functions and procedures owned by the current user

## USER_PROXIES

USER_PROXIES displays information about connections the current user is allowed to proxy. Its columns (except for PROXY) are the same as those in DBA_PROXIES.

**See Also:**

"DBA_PROXIES"

## USER_QUEUE_SCHEDULES

USER_QUEUE_SCHEDULES describes the propagation schedules whose source queues are owned by the current user. Its columns (except for SCHEMA) are the same as those in ALL_QUEUE_SCHEDULES.

**See Also:**

"ALL_QUEUE_SCHEDULES"

## USER_QUEUE_SUBSCRIBERS

USER_QUEUE_SUBSCRIBERS displays the list of subscribers on queues that are under the current user's schema. Its columns (except for OWNER) are the same as those in "ALL_QUEUE_SUBSCRIBERS".

## USER_QUEUE_TABLES

USER_QUEUE_TABLES describes the queues in the queue tables created in the current user's schema. Its columns (except for OWNER) are the same as those in ALL_QUEUE_TABLES.

**See Also:**

"ALL_QUEUE_TABLES"

## USER_QUEUES

USER_QUEUES describes the operational characteristics of every queue in the user's schema. Its columns (except for OWNER) are the same as those in "ALL_QUEUES".

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about these views and Advanced Queuing

## USER_RECYCLEBIN

USER_RECYCLEBIN displays information about the recycle bin owned by the current user. Its columns (except for OWNER) are the same as those in DBA_RECYCLEBIN.

**See Also:**

"DBA_RECYCLEBIN"

## USER_REFRESH

USER_REFRESH describes all refresh groups owned by the current user. Its columns are the same as those in "ALL_REFRESH".

## USER_REFRESH_CHILDREN

USER_REFRESH_CHILDREN lists all the objects in refresh groups owned by the current user. Its columns are the same as those in "ALL_REFRESH_CHILDREN".

## USER_REFS

USER_REFS describes the REF columns and REF attributes in the object type columns of tables owned by the current user. Its columns are the same as those in "ALL_REFS".

## USER_REGISTERED_MVIEWS

USER_REGISTERED_MVIEWS describes all registered materialized views (registered at a master site or a master materialized view site) owned by the current user. Its columns are the same as those in ALL_REGISTERED_MVIEWS.

**See Also:**

"ALL_REGISTERED_MVIEWS"

## USER_REGISTRY

USER_REGISTRY displays information about the components owned by the current user that are loaded into the component registry. Its columns are the same as those in DBA_REGISTRY.

**See Also:**

"DBA_REGISTRY"

## USER_RESOURCE_LIMITS

USER_RESOURCE_LIMITS displays the resource limits for the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| RESOURCE_NAME | VARCHAR2(32) | NOT NULL | Name of the resource |
| LIMIT | VARCHAR2(40) | | Limit placed on this resource |

## USER_RESUMABLE

USER_RESUMABLE displays the resumable statements executed by the current user. Its columns (except for USER_ID) are the same as those in DBA_RESUMABLE.

**See Also:**

"DBA_RESUMABLE"

## USER_REWRITE_EQUIVALENCES

USER_REWRITE_EQUIVALENCES describes the rewrite equivalences owned by the current user. Its columns are the same as those in ALL_REWRITE_EQUIVALENCES.

**See Also:**

"ALL_REWRITE_EQUIVALENCES"

## USER_ROLE_PRIVS

USER_ROLE_PRIVS describes the roles granted to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | | Name of the user, or PUBLIC |
| GRANTED_ROLE | VARCHAR2(128) | | Name of the role granted to the user |
| ADMIN_OPTION | VARCHAR2(3) | | Indicates whether the grant was with the ADMIN OPTION (YES) or not (NO) |
| DELEGATE_OPTION[1] | VARCHAR2(3) | | Indicates whether the grant was with the DELEGATE OPTION (YES) or not (NO) |
| DEFAULT_ROLE | VARCHAR2(3) | | Indicates whether the role is designated as a DEFAULT ROLE for the user (YES) or not (NO) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| OS_GRANTED | VARCHAR2(3) | | Indicates whether the role was granted by the operating system (YES) or not (NO); occurs if the OS_ROLES initialization parameter is true |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the role was granted commonly (CONTAINER=ALL was used)<br>• NO if the role was granted locally (CONTAINER=ALL was not used) |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DBA_ROLE_PRIVS"

# USER_RSRC_CONSUMER_GROUP_PRIVS

USER_RSRC_CONSUMER_GROUP_PRIVS displays information about the resource consumer groups to which the current user is assigned. Its columns (except for GRANTEE) are the same as those in DBA_RSRC_CONSUMER_GROUP_PRIVS.

**See Also:**

"DBA_RSRC_CONSUMER_GROUP_PRIVS"

# USER_RSRC_MANAGER_SYSTEM_PRIVS

USER_RSRC_MANAGER_SYSTEM_PRIVS displays information about the users who are granted system privileges for the DBMS_RESOURCE_MANAGER package. Its columns (except for GRANTEE) are the same as those in DBA_RSRC_MANAGER_SYSTEM_PRIVS.

**See Also:**

"DBA_RSRC_MANAGER_SYSTEM_PRIVS"

# USER_RULE_SET_RULES

USER_RULE_SET_RULES describes the rules in the rule sets owned by the current user. Its columns (except for RULE_SET_OWNER) are the same as those in ALL_RULE_SET_RULES.

**See Also:**

"ALL_RULE_SET_RULES"

## USER_RULE_SETS

USER_RULE_SETS describes the rule sets owned by the current user. Its columns (except for RULE_SET_OWNER) are the same as those in ALL_RULE_SETS.

**See Also:**

"ALL_RULE_SETS"

## USER_RULES

USER_RULES describes the rules owned by the current user. Its columns (except for RULE_OWNER) are the same as those in ALL_RULES.

**See Also:**

"ALL_RULES"

## USER_SCHEDULER_CHAIN_RULES

USER_SCHEDULER_CHAIN_RULES displays information about the rules for the chains owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_CHAIN_RULES.

**See Also:**

"ALL_SCHEDULER_CHAIN_RULES"

## USER_SCHEDULER_CHAIN_STEPS

USER_SCHEDULER_CHAIN_STEPS displays information about the defined steps of the chains owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_CHAIN_STEPS.

**See Also:**

"ALL_SCHEDULER_CHAIN_STEPS"

## USER_SCHEDULER_CHAINS

USER_SCHEDULER_CHAINS displays information about the chains owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_CHAINS.

**See Also:**

"ALL_SCHEDULER_CHAINS"

# USER_SCHEDULER_CREDENTIALS

USER_SCHEDULER_CREDENTIALS displays information about the credentials owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_CREDENTIALS.

**Note:**

This view is deprecated in favor of the USER_CREDENTIALS view. Oracle recommends that you use USER_CREDENTIALS instead. USER_SCHEDULER_CREDENTIALS is retained for backward compatibility only.

**See Also:**

- "USER_CREDENTIALS"
- "ALL_SCHEDULER_CREDENTIALS"

# USER_SCHEDULER_DB_DESTS

USER_SCHEDULER_DB_DESTS displays information about the destination objects owned by the current user pointing to remote databases. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_DB_DESTS.

**See Also:**

"ALL_SCHEDULER_DB_DESTS"

# USER_SCHEDULER_DESTS

USER_SCHEDULER_DESTS displays information about the destination objects for jobs owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_DESTS.

**See Also:**

"ALL_SCHEDULER_DESTS"

# USER_SCHEDULER_FILE_WATCHERS

USER_SCHEDULER_FILE_WATCHERS displays information about the Scheduler file watch requests owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_FILE_WATCHERS.

**See Also:**

"ALL_SCHEDULER_FILE_WATCHERS"

## USER_SCHEDULER_GROUP_MEMBERS

USER_SCHEDULER_GROUP_MEMBERS displays information about the members of the Scheduler object groups owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_GROUP_MEMBERS.

**See Also:**

"ALL_SCHEDULER_GROUP_MEMBERS"

## USER_SCHEDULER_GROUPS

USER_SCHEDULER_GROUPS displays information about the Scheduler object groups owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_GROUPS.

**See Also:**

"ALL_SCHEDULER_GROUPS"

## USER_SCHEDULER_JOB_ARGS

USER_SCHEDULER_JOB_ARGS displays information about the arguments of the Scheduler jobs owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_JOB_ARGS.

**See Also:**

"ALL_SCHEDULER_JOB_ARGS"

## USER_SCHEDULER_JOB_DESTS

USER_SCHEDULER_JOB_DESTS displays information about the state of the jobs owned by the current user at each of their destinations. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_JOB_DESTS.

**See Also:**

"ALL_SCHEDULER_JOB_DESTS"

## USER_SCHEDULER_JOB_LOG

USER_SCHEDULER_JOB_LOG displays log information for the Scheduler jobs owned by the current user. Its columns are the same as those in ALL_SCHEDULER_JOB_LOG.

**See Also:**

"ALL_SCHEDULER_JOB_LOG"

## USER_SCHEDULER_JOB_RUN_DETAILS

USER_SCHEDULER_JOB_RUN_DETAILS displays log run details for the Scheduler jobs owned by the current user. Its columns are the same as those in ALL_SCHEDULER_JOB_RUN_DETAILS.

**See Also:**

"ALL_SCHEDULER_JOB_RUN_DETAILS"

## USER_SCHEDULER_JOBS

USER_SCHEDULER_JOBS displays information about the Scheduler jobs owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_JOBS.

**See Also:**

"ALL_SCHEDULER_JOBS"

## USER_SCHEDULER_NOTIFICATIONS

USER_SCHEDULER_NOTIFICATIONS displays information about the E-mail notifications for the jobs owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_NOTIFICATIONS.

**See Also:**

"ALL_SCHEDULER_NOTIFICATIONS"

## USER_SCHEDULER_PROGRAM_ARGS

USER_SCHEDULER_PROGRAM_ARGS displays information about the arguments of the Scheduler programs owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_PROGRAM_ARGS.

**See Also:**

"ALL_SCHEDULER_PROGRAM_ARGS"

## USER_SCHEDULER_PROGRAMS

USER_SCHEDULER_PROGRAMS displays information about the Scheduler programs owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_PROGRAMS.

**See Also:**

"ALL_SCHEDULER_PROGRAMS"

## USER_SCHEDULER_REMOTE_JOBSTATE

USER_SCHEDULER_REMOTE_JOBSTATE displays information about the state of the jobs owned by the current user at remote databases. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_REMOTE_JOBSTATE.

**See Also:**

"ALL_SCHEDULER_REMOTE_JOBSTATE"

## USER_SCHEDULER_RUNNING_CHAINS

USER_SCHEDULER_RUNNING_CHAINS displays information about the chain steps of the running chains owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_RUNNING_CHAINS.

**See Also:**

"ALL_SCHEDULER_RUNNING_CHAINS"

## USER_SCHEDULER_RUNNING_JOBS

USER_SCHEDULER_RUNNING_JOBS displays information about the running Scheduler jobs owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_RUNNING_JOBS.

**See Also:**

"ALL_SCHEDULER_RUNNING_JOBS"

## USER_SCHEDULER_SCHEDULES

USER_SCHEDULER_SCHEDULES displays information about the Scheduler schedules owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SCHEDULER_SCHEDULES.

**See Also:**

"ALL_SCHEDULER_SCHEDULES"

## USER_SEC_RELEVANT_COLS

USER_SEC_RELEVANT_COLS describes the security relevant columns of the security policies for the tables and views owned by the current user. Its columns (except for OBJECT_OWNER) are the same as those in ALL_SEC_RELEVANT_COLS.

**See Also:**

"ALL_SEC_RELEVANT_COLS"

## USER_SECONDARY_OBJECTS

USER_SECONDARY_OBJECTS provides information about secondary objects associated with domain indexes owned by the current user. This view is only relevant in the context of domain indexes. Its columns are the same as those in "ALL_SECONDARY_OBJECTS".

## USER_SEGMENTS

USER_SEGMENTS describes the storage allocated for the segments owned by the current user's objects. Its columns (except for OWNER, HEADER_FILE, HEADER_BLOCK, and RELATIVE_FNO) are the same as those in DBA_SEGMENTS.

**See Also:**

"DBA_SEGMENTS"

## USER_SEQUENCES

USER_SEQUENCES describes all sequences owned by the current user. Its columns are the same as those in "ALL_SEQUENCES".

## USER_SOURCE

USER_SOURCE describes the text source of the stored objects owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_SOURCE".

## USER_SOURCE_AE

USER_SOURCE_AE describes the text source of the stored objects (across all editions) owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SOURCE_AE.

**See Also:**

"ALL_SOURCE_AE"

## USER_SQL_TRANSLATION_PROFILES

USER_SQL_TRANSLATION_PROFILES describes all SQL translation profiles owned by the user. Its columns (except for OWNER) are the same as those in ALL_SQL_TRANSLATION_PROFILES.

**See Also:**

"ALL_SQL_TRANSLATION_PROFILES"

## USER_SQL_TRANSLATIONS

USER_SQL_TRANSLATIONS describes all SQL translations owned by the user. Its columns (except for OWNER) are the same as those in ALL_SQL_TRANSLATIONS.

**See Also:**

"ALL_SQL_TRANSLATIONS"

## USER_SQLJ_TYPE_ATTRS

USER_SQLJ_TYPE_ATTRS describes the attributes of the SQLJ object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SQLJ_TYPE_ATTRS.

**See Also:**

"ALL_SQLJ_TYPE_ATTRS"

## USER_SQLJ_TYPE_METHODS

USER_SQLJ_TYPE_METHODS describes the methods of the SQLJ object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SQLJ_TYPE_METHODS.

**See Also:**

"ALL_SQLJ_TYPE_METHODS"

## USER_SQLJ_TYPES

USER_SQLJ_TYPES describes the SQLJ object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_SQLJ_TYPES.

**See Also:**

"ALL_SQLJ_TYPES"

## USER_SQLSET

USER_SQLSET displays information about the SQL tuning sets owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_SQLSET".

## USER_SQLSET_BINDS

USER_SQLSET_BINDS displays the bind values associated with the SQL tuning sets owned by the current user. Its columns (except for SQLSET_OWNER) are the same as those in "ALL_SQLSET_BINDS".

## USER_SQLSET_PLANS

USER_SQLSET_PLANS describes captured plans for statements in the SQL tuning sets owned by the current user. Its columns (except for SQLSET_OWNER) are the same as those in "ALL_SQLSET_PLANS".

## USER_SQLSET_REFERENCES

USER_SQLSET_REFERENCES describes whether or not the SQL tuning sets owned by the current user are active. Its columns (except for SQLSET_OWNER) are the same as those in "ALL_SQLSET_REFERENCES".

## USER_SQLSET_STATEMENTS

USER_SQLSET_STATEMENTS displays information about the SQL statements, along with their statistics, that form the SQL tuning sets owned by the current user. Its columns (except for SQLSET_OWNER) are the same as those in "ALL_SQLSET_STATEMENTS".

## USER_SQLTUNE_BINDS

USER_SQLTUNE_BINDS displays the bind values associated with the tuned SQL statements owned by the current user. Its columns are the same as those in DBA_SQLTUNE_BINDS.

**See Also:**

"DBA_SQLTUNE_BINDS"

## USER_SQLTUNE_PLANS

USER_SQLTUNE_PLANS displays information about the execution plans generated for the SQL statements owned by the current user during a SQL tuning session. Its columns are the same as those in DBA_SQLTUNE_PLANS.

**See Also:**

"DBA_SQLTUNE_PLANS"

## USER_SQLTUNE_RATIONALE_PLAN

USER_SQLTUNE_RATIONALE_PLAN displays the association between rationales and operations in the execution plan of the SQL statements owned by the current user. Its columns are the same as those in DBA_SQLTUNE_RATIONALE_PLAN.

**See Also:**

"DBA_SQLTUNE_RATIONALE_PLAN"

## USER_SQLTUNE_STATISTICS

USER_SQLTUNE_STATISTICS displays statistics associated with the SQL statements owned by the current user. Its columns are the same as those in DBA_SQLTUNE_STATISTICS.

**See Also:**

"DBA_SQLTUNE_STATISTICS"

## USER_SR_GRP_STATUS

USER_SR_GRP_STATUS provides information on the current refresh operations for the current synchronous refresh groups in the database which are owned by the current user. Its columns are the same as those in DBA_SR_GRP_STATUS.

**See Also:**

"DBA_SR_GRP_STATUS"

## USER_SR_GRP_STATUS_ALL

USER_SR_GRP_STATUS_ALL provides information on the refresh operations on the synchronous refresh groups in the database which are owned by the current user. Its columns are the same as those in DBA_SR_GRP_STATUS_ALL.

**See Also:**

"DBA_SR_GRP_STATUS_ALL"

## USER_SR_OBJ

USER_SR_OBJ provides information on the objects currently registered for synchronous refresh for current groups for the current user. Its columns are the same as those in DBA_SR_OBJ.

**See Also:**

"DBA_SR_OBJ"

## USER_SR_OBJ_ALL

USER_SR_OBJ_ALL provides information on the objects registered for synchronous refresh for current and defunct groups for the current user. Its columns are the same as those in DBA_SR_OBJ_ALL.

**See Also:**

"DBA_SR_OBJ_ALL"

## USER_SR_OBJ_STATUS

USER_SR_OBJ_STATUS provides information on the status of objects registered for synchronous refresh for the current refresh operations for the current synchronous refresh groups in the database which are owned by the current user. Its columns are the same as those in DBA_SR_OBJ_STATUS.

**See Also:**

"DBA_SR_OBJ_STATUS"

## USER_SR_OBJ_STATUS_ALL

USER_SR_OBJ_STATUS_ALL provides information on the status of objects registered for synchronous refresh in the database which are owned by the current user. Its columns are the same as those in DBA_SR_OBJ_STATUS_ALL.

**See Also:**

"DBA_SR_OBJ_STATUS_ALL"

## USER_SR_PARTN_OPS

USER_SR_PARTN_OPS provides information on the partition operations registered on the base tables of the materialized views registered for synchronous refresh belonging to the current user. Its columns are the same as those in DBA_SR_PARTN_OPS.

**See Also:**

"DBA_SR_PARTN_OPS"

## USER_SR_STLOG_EXCEPTIONS

USER_SR_STLOG_EXCEPTIONS provides information on the exceptions in the staging logs for the tables belonging to the current user processed by DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG. Its columns are the same as those in DBA_SR_STLOG_EXCEPTIONS.

**See Also:**

"DBA_SR_STLOG_EXCEPTIONS"

## USER_SR_STLOG_STATS

USER_SR_STLOG_STATS provides information on the statistics in the staging logs for the tables belonging to the current user processed by DBMS_SYNC_REFRESH.PREPARE_STAGING_LOG. Its columns are the same as those in DBA_SR_STLOG_STATS.

**See Also:**

"DBA_SR_STLOG_STATS"

## USER_STAT_EXTENSIONS

USER_STAT_EXTENSIONS displays information about the optimizer statistics extensions owned by the current user. Its columns (except for OWNER) are the same as those in ALL_STAT_EXTENSIONS.

**See Also:**

"ALL_STAT_EXTENSIONS"

## USER_STORED_SETTINGS

USER_STORED_SETTINGS lists information about the persistent parameter settings for stored PL/SQL units, but only shows information about PL/SQL units owned by the current user. USER_STORED_SETTINGS does not display the OWNER column. The rest of its columns are the same as those in "ALL_STORED_SETTINGS".

## USER_SUBPART_COL_STATISTICS

USER_SUBPART_COL_STATISTICS provides column statistics and histogram information for subpartitions of subpartitioned objects owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_SUBPART_COL_STATISTICS".

## USER_SUBPART_HISTOGRAMS

USER_SUBPART_HISTOGRAMS lists actual histogram data (end-points per histogram) for histograms on table subpartitions owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_SUBPART_HISTOGRAMS".

## USER_SUBPART_KEY_COLUMNS

USER_SUBPART_KEY_COLUMNS lists subpartitioning key columns for composite-partitioned tables (and local indexes on composite-partitioned tables) owned by the current user. Its columns are the same as those in ALL_SUBPART_KEY_COLUMNS.

**See Also:**

"ALL_SUBPART_KEY_COLUMNS"

## USER_SUBPARTITION_TEMPLATES

USER_SUBPARTITION_TEMPLATES describes the subpartition templates owned by the current user. Its columns (except for USER_NAME) are the same as those in ALL_SUBPARTITION_TEMPLATES.

**See Also:**

"ALL_SUBPARTITION_TEMPLATES"

## USER_SUBSCR_REGISTRATIONS

USER_SUBSCR_REGISTRATIONS displays information about the subscription registrations owned by the current user. Its columns are the same as those in DBA_SUBSCR_REGISTRATIONS.

**See Also:**

"DBA_SUBSCR_REGISTRATIONS"

## USER_SYNONYMS

USER_SYNONYMS describes the private synonyms (synonyms owned by the current user). Its columns (except for OWNER) are the same as those in ALL_SYNONYMS.

**See Also:**

"ALL_SYNONYMS"

## USER_SYS_PRIVS

USER_SYS_PRIVS describes system privileges granted to the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | | Name of the user, or PUBLIC |
| PRIVILEGE | VARCHAR2(40) | | System privilege |
| ADMIN_OPTION | VARCHAR2(3) | | Indicates whether the grant was with the ADMIN option (YES) or not (NO) |
| COMMON | VARCHAR2(3) | | Indicates how the grant was made. Possible values:<br>• YES if the privilege was granted commonly (CONTAINER=ALL was used)<br>• NO if the privilege was granted locally (CONTAINER=ALL was not used) |

**See Also:**

"DBA_SYS_PRIVS"

# USER_TAB_COL_STATISTICS

USER_TAB_COL_STATISTICS contains column statistics and histogram information extracted from "USER_TAB_COLUMNS". Its columns (except for OWNER) are the same as those in "ALL_TAB_COL_STATISTICS".

# USER_TAB_COLS

USER_TAB_COLS describes the columns of the tables, views, and clusters owned by the current user. This view differs from "USER_TAB_COLUMNS" in that system-generated hidden columns and invisible columns, which are user-generated hidden columns, are not filtered out. Its columns (except for OWNER) are the same as those in "ALL_TAB_COLS". To gather statistics for this view, use the DBMS_STATS package.

**See Also:**

*Oracle Database Administrator's Guide* for information about invisible columns

# USER_TAB_COLUMNS

USER_TAB_COLUMNS describes the columns of the tables, views, and clusters owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_TAB_COLUMNS". To gather statistics for this view, use the DBMS_STATS package.

The USER_TAB_COLS view can display system-generated hidden columns and invisible columns, which are user-generated hidden columns.

**See Also:**

*Oracle Database Administrator's Guide* for information about invisible columns

## USER_TAB_COMMENTS

USER_TAB_COMMENTS displays comments on the tables and views owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TAB_COMMENTS.

**See Also:**

"ALL_TAB_COMMENTS"

## USER_TAB_HISTGRM_PENDING_STATS

USER_TAB_HISTGRM_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_TAB_HISTGRM_PENDING_STATS".

## USER_TAB_HISTOGRAMS

USER_TAB_HISTOGRAMS describes histograms on columns of tables owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_TAB_HISTOGRAMS".

## USER_TAB_IDENTITY_COLS

USER_TAB_IDENTITY_COLS describes all table identity columns. Its columns (except for OWNER) are the same as those in ALL_TAB_IDENTITY_COLS.

**See Also:**

"ALL_TAB_IDENTITY_COLS"

## USER_TAB_MODIFICATIONS

USER_TAB_MODIFICATIONS describes modifications to all tables owned by the current user that have been modified since the last time statistics were gathered on the tables. Its columns are the same as those in "ALL_TAB_MODIFICATIONS".

**Note:**

This view is populated only for tables with the MONITORING attribute. It is intended for statistics collection over a long period of time. For performance reasons, the Oracle Database does not populate this view immediately when the actual modifications occur. Run the FLUSH_DATABASE_MONITORING_INFO procedure in the DBMS_STATS PL/SQL package to populate this view with the latest information. The ANALYZE_ANY system privilege is required to run this procedure.

## USER_TAB_PARTITIONS

USER_TAB_PARTITIONS describes partition-level partitioning information, partition storage parameters, and partition statistics generated by the DBMS_STATS package for all partitions owned by the current user. Its columns are the same as those in "ALL_TAB_PARTITIONS".

## USER_TAB_PENDING_STATS

USER_TAB_PENDING_STATS describes pending statistics for tables, partitions, and subpartitions owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_TAB_PENDING_STATS".

## USER_TAB_PRIVS

USER_TAB_PRIVS describes the object grants for which the current user is the object owner, grantor, or grantee. Its columns are the same as those in DBA_TAB_PRIVS.

**See Also:**

"DBA_TAB_PRIVS"

## USER_TAB_PRIVS_MADE

USER_TAB_PRIVS_MADE describes the object grants for which the current user is the object owner. Its columns (except for OWNER) are the same as those in ALL_TAB_PRIVS_MADE.

**See Also:**

"ALL_TAB_PRIVS_MADE"

## USER_TAB_PRIVS_RECD

USER_TAB_PRIVS_RECD describes the object grants for which the current user is the grantee. Its columns (except for GRANTEE) are the same as those in ALL_TAB_PRIVS_RECD.

**See Also:**

"ALL_TAB_PRIVS_RECD"

## USER_TAB_STAT_PREFS

USER_TAB_STAT_PREFS displays information about statistics preferences for the tables owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_TAB_STAT_PREFS".

## USER_TAB_STATISTICS

USER_TAB_STATISTICS displays optimizer statistics for the tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TAB_STATISTICS.

**See Also:**

"ALL_TAB_STATISTICS"

## USER_TAB_STATS_HISTORY

USER_TAB_STATS_HISTORY provides a history of table statistics modifications for all tables owned by the current user. Its columns (except for OWNER) are the same as those in "ALL_TAB_STATS_HISTORY".

## USER_TAB_SUBPARTITIONS

USER_TAB_SUBPARTITIONS describes, for each table subpartition owned by the current user, the subpartition name, name of the table and partition to which it belongs, and its storage attributes. Its columns are the same as those in "ALL_TAB_SUBPARTITIONS".

## USER_TABLES

USER_TABLES describes the relational tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TABLES. To gather statistics for this view, use the DBMS_STATS package.

**See Also:**

"ALL_TABLES"

## USER_TABLESPACES

USER_TABLESPACES describes the tablespaces accessible to the current user. Its columns (except for PLUGGED_IN) are the same as those in DBA_TABLESPACES.

**See Also:**

"DBA_TABLESPACES"

## USER_TRANSFORMATIONS

USER_TRANSFORMATIONS displays information about the transformations owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TRANSFORMATIONS.

**See Also:**

"ALL_TRANSFORMATIONS"

## USER_TRIGGER_COLS

USER_TRIGGER_COLS describes the use of columns in the triggers owned by the current user and in triggers on tables owned by the current user. Its columns are the same as those in ALL_TRIGGER_COLS.

**See Also:**

"ALL_TRIGGER_COLS"

## USER_TRIGGER_ORDERING

USER_TRIGGER_ORDERING describes the triggers owned by the current user that have FOLLOWS or PRECEDES ordering. Its columns (except for TRIGGER_OWNER) are the same as those in ALL_TRIGGER_ORDERING.

**See Also:**

"ALL_TRIGGER_ORDERING"

## USER_TRIGGERS

USER_TRIGGERS describes the triggers owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TRIGGERS.

**See Also:**

"ALL_TRIGGERS"

## USER_TS_QUOTAS

USER_TS_QUOTAS contains information about tablespace quotas for the current user. Its columns (except for USERNAME) the same as those in "DBA_TS_QUOTAS".

## USER_TSTZ_TAB_COLS

USER_TSTZ_TAB_COLS displays information about the columns of the tables owned by the current user, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types. Its columns (except for OWNER, COLUMN_NAME, NESTED, and VIRTUAL_COLUMN) are the same as those in ALL_TSTZ_TAB_COLS.

**See Also:**

"ALL_TSTZ_TAB_COLS"

## USER_TSTZ_TABLES

USER_TSTZ_TABLES displays information about the tables owned by the current user, which have columns defined on TIMESTAMP WITH TIME ZONE data types or object types containing attributes of TIMESTAMP WITH TIME ZONE data types. Its columns (except for OWNER) are the same as those in ALL_TSTZ_TABLES.

**See Also:**

"ALL_TSTZ_TABLES"

## USER_TUNE_MVIEW

USER_TUNE_MVIEW displays the result of executing the DBMS_ADVISOR.TUNE_MVIEW procedure. Its columns (except for OWNER) are the same as those in DBA_TUNE_MVIEW.

**See Also:**

"DBA_TUNE_MVIEW"

## USER_TYPE_ATTRS

USER_TYPE_ATTRS describes the attributes of the object types owned by the current user. Its columns (except for OWNER and CHAR_USED) are the same as those in ALL_TYPE_ATTRS.

**See Also:**

"ALL_TYPE_ATTRS"

## USER_TYPE_METHODS

USER_TYPE_METHODS describes the methods of the object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TYPE_METHODS.

**See Also:**

"ALL_TYPE_METHODS"

## USER_TYPE_VERSIONS

USER_TYPE_VERSIONS describes the versions of the object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TYPE_VERSIONS.

**See Also:**

"ALL_TYPE_VERSIONS"

## USER_TYPES

USER_TYPES describes the object types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_TYPES.

**See Also:**

"ALL_TYPES"

## USER_UNUSED_COL_TABS

USER_UNUSED_COL_TABS describes the tables owned by the current user that contain unused columns. Its columns (except for OWNER) are the same as those in ALL_UNUSED_COL_TABS.

**See Also:**

"ALL_UNUSED_COL_TABS"

## USER_UPDATABLE_COLUMNS

USER_UPDATABLE_COLUMNS describes columns in a join view that can be updated by the current user, subject to appropriate privileges. Its columns are the same as those in "ALL_UPDATABLE_COLUMNS".

**See Also:**

*Oracle Database Concepts* for information on updatable join views

## USER_USERS

USER_USERS describes the current user.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USERNAME | VARCHAR2(128) | NOT NULL | Name of the user |
| USER_ID | NUMBER | NOT NULL | ID number of the user |
| ACCOUNT_STATUS | VARCHAR2(32) | NOT NULL | Account status:<br>• OPEN<br>• EXPIRED<br>• EXPIRED(GRACE)<br>• LOCKED(TIMED)<br>• LOCKED<br>• EXPIRED & LOCKED(TIMED)<br>• EXPIRED(GRACE) & LOCKED(TIMED) |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| | | | • EXPIRED & LOCKED<br>• EXPIRED(GRACE) & LOCKED |
| LOCK_DATE | DATE | | Date the account was locked if account status was LOCKED |
| EXPIRY_DATE | DATE | | Date of expiration of the account |
| DEFAULT_TABLESPACE | VARCHAR2(30) | NOT NULL | Default tablespace for data |
| TEMPORARY_TABLESPACE | VARCHAR2(30) | NOT NULL | Name of the default tablespace for temporary tables or the name of a tablespace group |
| CREATED | DATE | NOT NULL | User creation date |
| INITIAL_RSRC_CONSUMER_GROUP | VARCHAR2(128) | | Initial resource consumer group for the user |
| EXTERNAL_NAME | VARCHAR2(4000) | | User external name |
| PROXY_ONLY_CONNECT | VARCHAR2(1) | | Indicates whether a user can connect directly (N) or whether the account can only be proxied (Y) by users who have proxy privileges for this account (that is, by users who have been granted the "connect through" privilege for this account).<br>For more information about creating proxy user accounts and authorizing users to connect through them, see *Oracle Database Security Guide*. |
| COMMON | VARCHAR2(3) | | Indicates whether a given user is common. Possible values<br>• YES if a user is common<br>• NO if a user is local (not common) |
| ORACLE_MAINTAINED | VARCHAR2(1) | | Denotes whether the user was created, and is maintained, by Oracle-supplied scripts (such as catalog.sql or catproc.sql). A user for which this column has the value Y must not be changed in any way except by running an Oracle-supplied script. |

**See Also:**

"DBA_USERS"

# USER_USTATS

USER_USTATS describes the user-defined statistics collected on the tables and indexes owned by the current user. Its columns are the same as those in ALL_USTATS.

**See Also:**

"ALL_USTATS"

# USER_VARRAYS

USER_VARRAYS describes the varrays owned by the current user. Its columns (except for OWNER) are the same as those in ALL_VARRAYS.

**See Also:**

"ALL_VARRAYS"

# USER_VIEWS

USER_VIEWS describes the views owned by the current user. Its columns (except for OWNER) are the same as those in ALL_VIEWS.

**See Also:**

"ALL_VIEWS"

# USER_VIEWS_AE

USER_VIEWS_AE describes the views (across all editions) owned by the current user. Its columns (except for OWNER) are the same as those in ALL_VIEWS_AE.

**See Also:**

"ALL_VIEWS_AE"

# USER_WALLET_ACES

USER_WALLET_ACES describes the status of access control entries for the current user to access wallets through PL/SQL network utility packages. Its columns (except for ACE_ORDER, START_DATE, END_DATE, GRANT_TYPE, INVERTED_PRINCIPAL, PRINCIPAL, PRINCIPAL_TYPE, and STATUS) are the same as those in DBA_WALLET_ACES.

**See Also:**

"DBA_WALLET_ACES"

# USER_WARNING_SETTINGS

USER_WARNING_SETTINGS displays information about the warning parameter settings for the objects owned by the current user. Its columns (except for OWNER) are the same as those in ALL_WARNING_SETTINGS.

**See Also:**

"ALL_WARNING_SETTINGS"

## USER_XML_INDEXES

USER_XML_INDEXES describes the XML indexes owned by the current user. Its columns (except for INDEX_OWNER) are the same as those in ALL_XML_INDEXES.

**See Also:**

"ALL_XML_INDEXES"

## USER_XML_NESTED_TABLES

USER_XML_NESTED_TABLES describes all the tables and their corresponding nested tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_NESTED_TABLES.

**See Also:**

"ALL_XML_NESTED_TABLES"

## USER_XML_OUT_OF_LINE_TABLES

USER_XML_OUT_OF_LINE_TABLES describes all the out of line tables connected to a given root table for the same schema owned by the current user. Its columns (except for TABLE_OWNER) are the same as those in ALL_XML_OUT_OF_LINE_TABLES.

**See Also:**

"ALL_XML_OUT_OF_LINE_TABLES"

## USER_XML_SCHEMA_ATTRIBUTES

USER_XML_SCHEMA_ATTRIBUTES describes all the attributes and their properties owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMA_ATTRIBUTES.

**See Also:**

"ALL_XML_SCHEMA_ATTRIBUTES"

## USER_XML_SCHEMA_COMPLEX_TYPES

USER_XML_SCHEMA_COMPLEX_TYPES describes all complex types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMA_COMPLEX_TYPES.

**See Also:**

"ALL_XML_SCHEMA_COMPLEX_TYPES"

## USER_XML_SCHEMA_ELEMENTS

USER_XML_SCHEMA_ELEMENTS describes all the elements and their properties owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMA_ELEMENTS.

**See Also:**

"ALL_XML_SCHEMA_ELEMENTS"

## USER_XML_SCHEMA_NAMESPACES

USER_XML_SCHEMA_NAMESPACES describes all the available namespaces owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMA_NAMESPACES.

**See Also:**

"ALL_XML_SCHEMA_NAMESPACES"

## USER_XML_SCHEMA_SIMPLE_TYPES

USER_XML_SCHEMA_SIMPLE_TYPES describes all simple types owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMA_SIMPLE_TYPES.

**See Also:**

"ALL_XML_SCHEMA_SIMPLE_TYPES"

## USER_XML_SCHEMA_SUBSTGRP_HEAD

USER_XML_SCHEMA_SUBSTGRP_HEAD describes the heads of substitution groups owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMA_SUBSTGRP_HEAD.

**See Also:**

"ALL_XML_SCHEMA_SUBSTGRP_HEAD"

## USER_XML_SCHEMA_SUBSTGRP_MBRS

USER_XML_SCHEMA_SUBSTGRP_MBRS describes all members of substitution groups owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMA_SUBSTGRP_MBRS.

**See Also:**

"ALL_XML_SCHEMA_SUBSTGRP_MBRS"

## USER_XML_SCHEMAS

USER_XML_SCHEMAS describes the registered XML schemas owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_SCHEMAS.

**See Also:**

"ALL_XML_SCHEMAS"

## USER_XML_TAB_COLS

USER_XML_TAB_COLS describes the columns of the XML tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_TAB_COLS.

**See Also:**

"ALL_XML_TAB_COLS"

## USER_XML_TABLES

USER_XML_TABLES describes the XML tables owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_TABLES.

**See Also:**

"ALL_XML_TABLES"

## USER_XML_VIEW_COLS

USER_XML_VIEW_COLS describes the columns of the XML views owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_VIEW_COLS.

**See Also:**

"ALL_XML_VIEW_COLS"

# USER_XML_VIEWS

USER_XML_VIEWS describes the XML views owned by the current user. Its columns (except for OWNER) are the same as those in ALL_XML_VIEWS.

**See Also:**

"ALL_XML_VIEWS"

# USER_ZONEMAP_MEASURES

USER_ZONEMAP_MEASURES describes the measures for all the zone maps owned by the user. Its columns are the same as those in ALL_ZONEMAP_MEASURES.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2), and is intended for use with Oracle Exadata release 12.1.2.1.1 or later.

**See Also:**

- "ALL_ZONEMAP_MEASURES"
- *Oracle Database Data Warehousing Guide* for more information about zone maps

# USER_ZONEMAPS

USER_ZONEMAPS describes the zone maps owned by the user. Its columns are the same as those in ALL_ZONEMAPS.

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2), and is intended for use with Oracle Exadata release 12.1.2.1.1 or later.

**See Also:**

- "ALL_ZONEMAPS"
- *Oracle Database Data Warehousing Guide* for more information about zone maps

# Part III

## Dynamic Performance Views

This part describes the dynamic performance views, which are often referred to as V$ views. This part contains the following chapters:


- Dynamic Performance (V$) Views: V$ACCESS to V$HVMASTER_INFO
- Dynamic Performance (V$) Views: V$IM_COLUMN_LEVEL to V$RULE_SET_AGGREGATE_STATS
- Dynamic Performance (V$) Views: V$SCHEDULER_RUNNING_JOBS to V$XSTREAM_TRANSACTION

# 7

# Dynamic Performance (V$) Views: V$ACCESS to V$HVMASTER_INFO

This chapter describes the first set (in alphabetical order) of dynamic performance views. The remaining dynamic performance views appear in alphabetical order in Dynamic Performance (V$) Views: V$IM_COLUMN_LEVEL to V$RULE_SET_AGGREGATE_STATS through Dynamic Performance (V$) Views: V$SCHEDULER_RUNNING_JOBS to V$XSTREAM_TRANSACTION.

This chapter contains the following topics:


- About Dynamic Performance Views
- Dynamic Performance View Descriptions


## About Dynamic Performance Views

Oracle contains a set of underlying views that are maintained by the database server and accessible to the database administrator user SYS. These views are called **dynamic performance views** because they are continuously updated while a database is open and in use, and their contents relate primarily to performance.

Although these views appear to be regular database tables, they are not. These views provide data on internal disk structures and memory structures. You can select from these views, but you can never update or alter them.

---

**Note:**

- You can query the dynamic performance views to extract information from them. However, only simple queries are supported. If sorts, joins, GROUP BY clauses and the like are needed, then you should copy the information from each V$ view into a table (for example, using a CREATE TABLE ... AS SELECT statement), and then query from those tables.
- Because the information in the V$ views is dynamic, read consistency is not guaranteed for SELECT operations on these views.

---

The `catalog.sql` script contains definitions of the views and public synonyms for the dynamic performance views. You must run `catalog.sql` to create these views and synonyms. After installation, only user SYS or anyone with SYSDBA privilege has access to the dynamic performance tables. See *Oracle Database Administrator's Guide* for more information about running `catalog.sql`.

## V$ Views

The actual dynamic performance views are identified by the prefix V_$. Public synonyms for these views have the prefix V$. Database administrators and other users should access only the V$ objects, not the V_$ objects.

The dynamic performance views are used by Oracle Enterprise Manager, which is the primary interface for accessing information about system performance. After an instance is started, the V$ views that read from memory are accessible. Views that read data from disk require that the database be mounted, and some require that the database be open.

V$ views are CONTAINER_DATA objects. When a user connected to the root queries a V $ view, the query results will depend on the CONTAINER_DATA attribute for users for the view. The CONTAINER_DATA clause of the SQL ALTER USER statement is used to set and modify users' CONTAINER_DATA attribute.

**See Also:**

- *Oracle Database Security Guide* for more information about container data objects
- *Oracle Database SQL Language Reference* for more information about the CONTAINER_DATA clause for the SQL ALTER USER statement

## GV$ Views

For almost every V$ view described in this chapter, Oracle has a corresponding GV$ (global V$) view. In Oracle Real Application Clusters, querying a GV$ view retrieves the V$ view information from all qualified instances. In addition to the V$ information, each GV$ view contains an extra column named INST_ID of data type NUMBER. The INST_ID column displays the instance number from which the associated V$ view information was obtained. The INST_ID column can be used as a filter to retrieve V$ information from a subset of available instances. For example, the following query retrieves the information from the V$LOCK view on instances 2 and 5:

```
SQL> SELECT * FROM GV$LOCK WHERE INST_ID = 2 OR INST_ID = 5;
```

**See Also:**

*Oracle Real Application Clusters Installation and Configuration Guide* for your operating system

GV$ views are CONTAINER_DATA objects. When a user connected to the root queries a GV$ view, the query results will depend on the CONTAINER_DATA attribute for users for the view. The CONTAINER_DATA clause of the SQL ALTER USER statement is used to set and modify users' CONTAINER_DATA attribute.

**See Also:**

- *Oracle Database Security Guide* for more information about container data objects
- *Oracle Database SQL Language Reference* for more information about the `CONTAINER_DATA` clause for the SQL `ALTER USER` statement

## Dynamic Performance View Descriptions

The remainder of this chapter describes the dynamic performance views in alphabetical order.

## V$ACCESS

`V$ACCESS` displays information about locks that are currently imposed on library cache objects. The locks are imposed to ensure that they are not aged out of the library cache while they are required for SQL execution.

| Column | Datatype | Description |
|---|---|---|
| `SID` | `NUMBER` | Session number that is accessing an object |
| `OWNER` | `VARCHAR2(64)` | Owner of the object |
| `OBJECT` | `VARCHAR2(1000)` | Name of the object |
| `TYPE` | `VARCHAR2(64)` | Type identifier for the object |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$ACTIVE_INSTANCES

`V$ACTIVE_INSTANCES` displays the mapping between instance names and instance numbers for all instances that have the database currently mounted.

| Column | Datatype | Description |
|---|---|---|
| `INST_NUMBER` | `NUMBER` | Instance number |
| `INST_NAME` | `VARCHAR2(60)` | Instance name |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ACTIVE_SERVICES

V$ACTIVE_SERVICES displays information about the active services in the database.

| Column | Datatype | Description |
|---|---|---|
| SERVICE_ID | NUMBER | Service ID |
| NAME | VARCHAR2(64) | Name of the service |
| NAME_HASH | NUMBER | Service name hash |
| NETWORK_NAME | VARCHAR2(512) | Network name |
| CREATION_DATE | DATE | Creation date |
| CREATION_DATE_HASH | NUMBER | Creation date hash |
| GOAL | VARCHAR2(12) | Runtime Load Balancing Goal being used to create run-time load balancing and connection load balancing advice:<br>• NONE<br>• SERVICE_TIME - Connections are balanced by response time<br>• THROUGHPUT - Connections are balanced by throughput |
| DTP | VARCHAR2(1) | Indicates whether or not Distributed Transaction Processing is enabled for this service. When Distributed Transaction Processing is set to Y (YES), it means that the service is offered at exactly one instance at a time for XA affinity. Possible values:<br>• Y - YES<br>• N - NO |
| BLOCKED | VARCHAR2(3) | Indicates whether a service on the specified instance is blocked from accepting new connections altogether (YES) or not (NO). If a service is blocked, then all connections will be directed to other instances (if any) that are hosting the desired service. |
| AQ_HA_NOTIFICATION | VARCHAR2(3) | Indicates whether FAN - Fast Application Notification for OCI connections is set (YES) or not (NO) |
| CLB_GOAL | VARCHAR2(5) | Connection load balancing goal used with statistics that are sent to the listeners to determine how new connections are distributed:<br>• LONG<br>• SHORT |
| COMMIT_OUTCOME | VARCHAR2(3) | For Transaction Guard, indicates whether the database service associated with the user session has the COMMIT_OUTCOME service attribute enabled (YES) or not (NO).<br>When the attribute is enabled:<br>• The outcome of a COMMIT transaction is durable, which means the status of the last COMMIT executed can be looked up after an outage. |

| Column | Datatype | Description |
|---|---|---|
| | | • A logical transaction ID (LTXID) is set for each user session at session creation and successful commit.<br>**See Also:** For information about preserving the commit outcome, see *Oracle Database Development Guide*. For information about logical transaction IDs, see *Oracle Database Development Guide* |
| RETENTION_TIME | NUMBER | For Transaction Guard (COMMIT_OUTCOME set to TRUE), this parameter determines the amount of time (in seconds) that the commit outcome is retained in the database |
| REPLAY_INITIATION_TIMEOUT | NUMBER | For Application Continuity, this option specifies the difference between the time (in seconds) of original execution of the first operation of a request and the time that the replay is ready to start after a successful reconnect. Application Continuity will not replay after the specified amount of time has passed. This option is intended to avoid the unintentional execution of a transaction when a system is recovered after a long period of time. The default is 5 minutes (300 seconds). |
| SESSION_STATE_CONSISTENCY | VARCHAR2(30) | Describes how non-transactional is changed during a request. This parameter is considered only if failover_type is set to TRANSACTION for Application Continuity. Examples of session state are NLS settings, optimizer preferences, event settings, PL/SQL global variables, temporary tables, advanced queues, LOBs, and result cache. If non-transactional values change after the request starts, the default value of DYNAMIC should be set. Almost all applications should use DYNAMIC mode. If you are unsure, use DYNAMIC mode. |
| GLOBAL | VARCHAR2(3) | Indicates whether the service is global. A global service is managed by Global Service Manager (GSM) and can be provided by multiple databases that contain replicated data. Possible values:<br>• YES: Indicates the service is global<br>• NO: Indicates the service is not global |
| CON_NAME | VARCHAR2(30) | Container name of the object. The value of this column is NULL in non-CDBs. |
| SQL_TRANSLATION_PROFILE | VARCHAR2(65) | A non-NULL value specifies the initial SQL translation profile for subsequent database connections that use the service and do not specify a SQL translation profile. A NULL value has no effect. |
| MAX_LAG_TIME | VARCHAR2(30) | The maximum replication lag (in seconds) that is acceptable for a data replica to be used for providing the database service. Can only be specified for global services. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ACTIVE_SESS_POOL_MTH

V$ACTIVE_SESS_POOL_MTH displays available active session pool resource allocation methods.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(40) | Name of the active session pool resource allocation method |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ACTIVE_SESSION_HISTORY

V$ACTIVE_SESSION_HISTORY displays sampled session activity in the database. It contains snapshots of active database sessions taken once a second. A database session is considered active if it was on the CPU or was waiting for an event that didn't belong to the Idle wait class. Refer to the V$EVENT_NAME view for more information on wait classes.

This view contains one row for each active session per sample and returns the latest session sample rows first. A majority of the columns describing the session in the active session history are present in the V$SESSION view.

| Column | Datatype | Description |
|---|---|---|
| SAMPLE_ID | NUMBER | ID of the sample |
| SAMPLE_TIME | TIMESTAMP(3) | Time at which the sample was taken |
| IS_AWR_SAMPLE | VARCHAR2(1) | Indicates whether this sample has been flushed or will be flushed to the Automatic Workload Repository (DBA_HIST_ACTIVE_SESS_HISTORY) (Y) or not (N) |
| SESSION_ID | NUMBER | Session identifier; maps to V$SESSION.SID |
| SESSION_SERIAL# | NUMBER | Session serial number (used to uniquely identify a session's objects); maps to V$SESSION.SERIAL# |
| SESSION_TYPE | VARCHAR2(10) | Session type:<br>• FOREGROUND<br>• BACKGROUND |
| FLAGS | NUMBER | Reserved for future use |
| USER_ID | NUMBER | Oracle user identifier; maps to V$SESSION.USER# |
| SQL_ID | VARCHAR2(13) | SQL identifier of the SQL statement that the session was executing at the time of sampling |
| IS_SQLID_CURRENT | VARCHAR2(1) | Indicates whether the SQL identifier in the SQL_ID column is being executed (Y) or not (N) |

| Column | Datatype | Description |
|---|---|---|
| SQL_CHILD_NUMBER | NUMBER | Child number of the SQL statement that the session was executing at the time of sampling |
| SQL_OPCODE | NUMBER | Indicates what phase of operation the SQL statement was in; maps to V$SESSION.COMMAND<br>**See Also:** "V$SESSION" for information on interpreting this column |
| SQL_OPNAME | VARCHAR2(64) | SQL command name |
| FORCE_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to FORCE |
| TOP_LEVEL_SQL_ID | VARCHAR2(13) | SQL identifier of the top level SQL statement |
| TOP_LEVEL_SQL_OPCODE | NUMBER | Indicates what phase of operation the top level SQL statement was in |
| SQL_ADAPTIVE_PLAN_RESOLVED | NUMBER | Indicates whether the SQL plan of the sampled database session is a resolved adaptive plan or not |
| SQL_FULL_PLAN_HASH_VALUE | NUMBER | Numerical representation of the complete SQL plan for the cursor being executed by this session |
| SQL_PLAN_HASH_VALUE | NUMBER | Numeric representation of the SQL plan for the cursor. This information might not be available for all session samples. V$SESSION does not contain this information. |
| SQL_PLAN_LINE_ID | NUMBER | SQL plan line ID |
| SQL_PLAN_OPERATION | VARCHAR2(30) | Plan operation name |
| SQL_PLAN_OPTIONS | VARCHAR2(30) | Plan operation options |
| SQL_EXEC_ID | NUMBER | SQL execution identifier |
| SQL_EXEC_START | DATE | Time when the execution of the SQL started |
| PLSQL_ENTRY_OBJECT_ID | NUMBER | Object ID of the top-most PL/SQL subprogram on the stack; NULL if there is no PL/SQL subprogram on the stack. Maps to DBA_OBJECTS.OBJECT_ID. |
| PLSQL_ENTRY_SUBPROGRAM_ID | NUMBER | Subprogram ID of the top-most PL/SQL subprogram on the stack. Maps to DBA_OBJECTS.DATA_OBJECT_ID. |
| PLSQL_OBJECT_ID | NUMBER | Object ID of the currently executing PL/SQL subprogram. Maps to DBA_OBJECTS.OBJECT_ID. |
| PLSQL_SUBPROGRAM_ID | NUMBER | Subprogram ID of the currently executing PL/SQL object; NULL if executing SQL. Maps to DBA_OBJECTS.DATA_OBJECT_ID. |
| QC_INSTANCE_ID | NUMBER | Query coordinator instance ID. This information is only available if the sampled session is a parallel query slave. For all other sessions, the value is 0. |
| QC_SESSION_ID | NUMBER | Query coordinator session ID. This information is only available if the sampled session is a parallel query slave. For all other sessions, the value is 0. |

| Column | Datatype | Description |
|---|---|---|
| QC_SESSION_SERIAL# | NUMBER | Query coordinator session serial number. This information is only available if the sampled session is a parallel query slave. For all other sessions, the value is 0. |
| PX_FLAGS | NUMBER | Reserved for internal use |
| EVENT | VARCHAR2(64) | If SESSION_STATE = WAITING, then the event for which the session was waiting for at the time of sampling.<br>If SESSION_STATE = ON CPU, then this column is NULL.<br>**See Also:** Oracle Wait Events |
| EVENT_ID | NUMBER | Identifier of the resource or event for which the session is waiting or for which the session last waited. Interpretation is similar to that of the EVENT column. |
| EVENT# | NUMBER | Number of the resource or event for which the session is waiting or for which the session last waited. Interpretation is similar to that of the EVENT column. |
| SEQ# | NUMBER | Sequence number that uniquely identifies the wait (incremented for each wait) |
| P1TEXT | VARCHAR2(64) | Text of the first additional parameter |
| P1 | NUMBER | First additional parameter |
| P2TEXT | VARCHAR2(64) | Text of the second additional parameter |
| P2 | NUMBER | Second additional parameter |
| P3TEXT | VARCHAR2(64) | Text of the third additional parameter |
| P3 | NUMBER | Third additional parameter |
| WAIT_CLASS | VARCHAR2(64) | Wait class name of the event for which the session was waiting at the time of sampling. Interpretation is similar to that of the EVENT column. Maps to V$SESSION.WAIT_CLASS. |
| WAIT_CLASS_ID | NUMBER | Wait class identifier of the event for which the session was waiting at the time of sampling. Interpretation is similar to that of the EVENT column. Maps to V$SESSION.WAIT_CLASS_ID. |
| WAIT_TIME | NUMBER | Total wait time for the event for which the session last waited if the session was on the CPU when sampled; 0 if the session was waiting at the time of sampling<br>**Note:** Whether or not WAIT_TIME = 0 is what is useful to find the SESSION_STATE at the time of sampling, rather than the actual value of WAIT_TIME itself. Maps to V $SESSION.WAIT_TIME. |
| SESSION_STATE | VARCHAR2(7) | Session state:<br>• WAITING<br>• ON CPU |
| TIME_WAITED | NUMBER | If SESSION_STATE = WAITING, then the time that the session actually spent waiting for that event (in microseconds). This column is set for waits that were in progress at the time the sample was taken. |

| Column | Datatype | Description |
|---|---|---|
| | | If a wait event lasted for more than a second and was caught waiting in more than one session sample row, then the actual time spent waiting for that wait event will be populated in the last of those session sample rows. At any given time, this information will not be available for the latest session sample. |
| BLOCKING_SESSION_STATUS | VARCHAR2(11) | Status of the blocking session:<br>• VALID<br>• NO HOLDER<br>• GLOBAL<br>• NOT IN WAIT<br>• UNKNOWN |
| BLOCKING_SESSION | NUMBER | Session identifier of the blocking session. Populated only if the blocker is on the same instance and the session was waiting for enqueues or a "buffer busy" wait. Maps to V$SESSION.BLOCKING_SESSION. |
| BLOCKING_SESSION_SERIAL# | NUMBER | Serial number of the blocking session |
| BLOCKING_INST_ID | NUMBER | Instance number of the blocker shown in BLOCKING_SESSION |
| BLOCKING_HANGCHAIN_INFO | VARCHAR2(1) | Indicates whether the information about BLOCKING_SESSION comes from the hang chain (Y) or not (N) |
| CURRENT_OBJ# | NUMBER | Object ID of the object that the session is referencing. This information is only available if the session was waiting for application, cluster, concurrency, and user I/O wait events. Maps to V$SESSION.ROW_WAIT_OBJ#. |
| CURRENT_FILE# | NUMBER | File number of the file containing the block that the session is referencing. This information is only available if the session was waiting for cluster, concurrency, and user I/O wait events. Maps to V$SESSION.ROW_WAIT_FILE#. |
| CURRENT_BLOCK# | NUMBER | ID of the block that the session is referencing. This information is only available if the session was waiting for cluster, concurrency, and user I/O wait events. Maps to V$SESSION.ROW_WAIT_BLOCK#. |
| CURRENT_ROW# | NUMBER | Row identifier that the session is referencing. This information is only available if the session was waiting for cluster, concurrency, and user I/O wait events. Maps to V$SESSION.ROW_WAIT_ROW#. |
| TOP_LEVEL_CALL# | NUMBER | Oracle top level call number |
| TOP_LEVEL_CALL_NAME | VARCHAR2(64) | Oracle top level call name |
| CONSUMER_GROUP_ID | NUMBER | Consumer group ID |
| XID | RAW(8) | Transaction ID that the session was working on at the time of sampling. V$SESSION does not contain this information. |
| REMOTE_INSTANCE# | NUMBER | Remote instance identifier that will serve the block that this session is waiting for. This information is only available if the session was waiting for cluster events. |

| Column | Datatype | Description |
|---|---|---|
| TIME_MODEL | NUMBER | Time model information |
| IN_CONNECTION_MGMT | VARCHAR2(1) | Indicates whether the session was doing connection management at the time of sampling (Y) or not (N) |
| IN_PARSE | VARCHAR2(1) | Indicates whether the session was parsing at the time of sampling (Y) or not (N) |
| IN_HARD_PARSE | VARCHAR2(1) | Indicates whether the session was hard parsing at the time of sampling (Y) or not (N) |
| IN_SQL_EXECUTION | VARCHAR2(1) | Indicates whether the session was executing SQL statements at the time of sampling (Y) or not (N) |
| IN_PLSQL_EXECUTION | VARCHAR2(1) | Indicates whether the session was executing PL/SQL at the time of sampling (Y) or not (N) |
| IN_PLSQL_RPC | VARCHAR2(1) | Indicates whether the session was executing inbound PL/SQL RPC calls at the time of sampling (Y) or not (N) |
| IN_PLSQL_COMPILATION | VARCHAR2(1) | Indicates whether the session was compiling PL/SQL at the time of sampling (Y) or not (N) |
| IN_JAVA_EXECUTION | VARCHAR2(1) | Indicates whether the session was executing Java at the time of sampling (Y) or not (N) |
| IN_BIND | VARCHAR2(1) | Indicates whether the session was doing bind operations at the time of sampling (Y) or not (N) |
| IN_CURSOR_CLOSE | VARCHAR2(1) | Indicates whether the session was closing a cursor at the time of sampling (Y) or not (N) |
| IN_SEQUENCE_LOAD | VARCHAR2(1) | Indicates whether the session is loading in sequence (in sequence load code) (Y) or not (N) |
| IN_INMEMORY_QUERY | VARCHAR2(1) | Indicates whether the session was querying the In-Memory Column Store (IM column store) at the time of sampling (Y) or not (N) |
| IN_INMEMORY_POPULATE[1] | VARCHAR2(1) | Indicates whether the session was populating the IM column store at the time of sampling (Y) or not (N) |
| IN_INMEMORY_PREPOPULATE[1] | VARCHAR2(1) | Indicates whether the session was prepopulating the IM column store at the time of sampling (Y) or not (N) |
| IN_INMEMORY_REPOPULATE[1] | VARCHAR2(1) | Indicates whether the session was repopulating the IM column store at the time of sampling (Y) or not (N) |
| IN_INMEMORY_TREPOPULATE[1] | VARCHAR2(1) | Indicates whether the session was trickle repopulating the IM column store at the time of sampling (Y) or not (N) |
| CAPTURE_OVERHEAD | VARCHAR2(1) | Indicates whether the session is executing capture code (Y) or not (N) |
| REPLAY_OVERHEAD | VARCHAR2(1) | Indicates whether the session is executing replay code (Y) or not (N) |
| IS_CAPTURED | VARCHAR2(1) | Indicates whether the session is being captured (Y) or not (N) |
| IS_REPLAYED | VARCHAR2(1) | Indicates whether the session is being replayed (Y) or not (N) |

| Column | Datatype | Description |
|---|---|---|
| SERVICE_HASH | NUMBER | Hash that identifies the Service; maps to V$ACTIVE_SERVICES.NAME_HASH |
| PROGRAM | VARCHAR2(48) | Name of the operating system program |
| MODULE | VARCHAR2(64) | Name of the executing module when sampled, as set by the DBMS_APPLICATION_INFO.SET_MODULE procedure |
| ACTION | VARCHAR2(64) | Name of the executing module when sampled, as set by the DBMS_APPLICATION_INFO.SET_ACTION procedure |
| CLIENT_ID | VARCHAR2(64) | Client identifier of the session; maps to V$SESSION.CLIENT_IDENTIFIER |
| MACHINE | VARCHAR2(64) | Client's operating system machine name |
| PORT | NUMBER | Client port number |
| ECID | VARCHAR2(64) | Execution context identifier (sent by Application Server) |
| DBREPLAY_FILE_ID | NUMBER | If the session is being captured or replayed, then DBREPLAY_FILE_ID is the file ID for the workload capture or workload replay; otherwise it is NULL. |
| DBREPLAY_CALL_COUNTER | NUMBER | If the session is being captured or replayed, then DBREPLAY_CALL_COUNTER is the call counter of the user call that is being captured or replayed; otherwise it is NULL. |
| TM_DELTA_TIME | NUMBER | Time interval (in microseconds) over which TM_DELTA_CPU_TIME and TM_DELTA_DB_TIME are accumulated |
| TM_DELTA_CPU_TIME | NUMBER | Amount of time this session spent on CPU over the last TM_DELTA_TIME microseconds |
| TM_DELTA_DB_TIME | NUMBER | Amount of time spent by this session in database calls over the last TM_DELTA_TIME microseconds |
| DELTA_TIME | NUMBER | Time interval (in microseconds) since the last time this session was sampled or created, over which the next five statistics are accumulated |
| DELTA_READ_IO_REQUESTS | NUMBER | Number of read I/O requests made by this session over the last DELTA_TIME microseconds |
| DELTA_WRITE_IO_REQUESTS | NUMBER | Number of write I/O requests made by this session over the last DELTA_TIME microseconds |
| DELTA_READ_IO_BYTES | NUMBER | Number of I/O bytes read by this session over the last DELTA_TIME microseconds |
| DELTA_WRITE_IO_BYTES | NUMBER | Number of I/O bytes written by this session over the last DELTA_TIME microseconds |
| DELTA_INTERCONNECT_IO_BYTES | NUMBER | Number of I/O bytes sent over the I/O interconnect over the last DELTA_TIME microseconds |
| DELTA_READ_MEM_BYTES | NUMBER | Number of read bytes through the buffer cache |

| Column | Datatype | Description |
|---|---|---|
| PGA_ALLOCATED | NUMBER | Amount of PGA memory (in bytes) consumed by this session at the time this sample was taken |
| TEMP_SPACE_ALLOCATED | NUMBER | Amount of TEMP memory (in bytes) consumed by this session at the time this sample was taken |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| DBOP_NAME | VARCHAR2(30) | Database operation name. If the type is SQL, the DBOP_NAME will be NULL. |
| DBOP_EXEC_ID | NUMBER | Database operation execution identifier for the current execution. If the type is SQL, the DBOP_EXEC_ID will be NULL. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "V$SESSION"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_MODULE procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_ACTION procedure

# V$ADVISOR_PROGRESS

V$ADVISOR_PROGRESS displays information about the progress of advisor execution.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID |
| SERIAL# | NUMBER | Session serial number |
| USERNAME | VARCHAR2(30) | Oracle user name |
| OPNAME | VARCHAR2(64) | Operation name |
| ADVISOR_NAME | VARCHAR2(64) | Advisor name |
| TASK_ID | NUMBER | Task ID |
| TARGET_DESC | VARCHAR2(32) | Description of the target of the advisor |
| SOFAR | NUMBER | Amount of work done so far |
| TOTALWORK | NUMBER | Total work to be done |

| Column | Datatype | Description |
|---|---|---|
| UNITS | VARCHAR2(32) | Units that the work is measured in |
| BENEFIT_SOFAR | NUMBER | Benefit obtained so far |
| BENEFIT_MAX | NUMBER | Estimate of maximum benefit that could be obtained |
| FINDINGS | NUMBER | Number of findings so far |
| RECOMMENDATIONS | NUMBER | Number of recommendations so far |
| TIME_REMAINING | NUMBER | Estimate of time remaining for the completion of the task (in seconds) |
| START_TIME | DATE | Start time of the task |
| LAST_UPDATE_TIME | DATE | Last time progress was posted |
| ELAPSED_SECONDS | NUMBER | Elapsed time so far |
| ADVISOR_METRIC1 | NUMBER | Value of the advisor-specific metric |
| METRIC1_DESC | VARCHAR2(64) | Description of the advisor-specific metric |
| EXECUTION_TYPE | VARCHAR2(64) | Type of the last execution. This information is optional for single-execution tasks. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ALERT_TYPES

V$ALERT_TYPES displays information about server alert types.

| Column | Datatype | Description |
|---|---|---|
| REASON_ID | NUMBER | ID of the alert reason |
| OBJECT_TYPE | VARCHAR2(64) | Object type |
| TYPE | VARCHAR2(9) | Alert type:<br>• Stateful<br>• Stateless |
| GROUP_NAME | VARCHAR2(64) | Group name |
| SCOPE | VARCHAR2(8) | Scope:<br>• Database<br>• Instance |
| INTERNAL_METRIC_CATEGORY | VARCHAR2(64) | Internal metric category |

| Column | Datatype | Description |
|---|---|---|
| INTERNAL_METRIC_NAME | VARCHAR2(64) | Internal metric name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$AQ

V$AQ displays statistics for the queues in the database.

| Column | Datatype | Description |
|---|---|---|
| QID | NUMBER | Unique queue identifier |
| WAITING | NUMBER | Number of messages in the queue in the state 'WAITING' |
| READY | NUMBER | Number of messages in the queue in the state 'READY' |
| EXPIRED | NUMBER | Number of messages in the queue the state 'EXPIRED' |
| AVERAGE_MSG_AGE | NUMBER | Average age of the messages in the queue' |
| TOTAL_WAIT | NUMBER | Total wait time of all 'READY' messages in the queue |
| AVERAGE_WAIT | NUMBER | Average wait time of 'READY' messages in the queue |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_BACKGROUND_COORDINATOR

V$AQ_BACKGROUND_COORDINATOR lists performance statistics for the Oracle Database Advanced Queueing master background coordinator process (AQPC).

| Column | Datatype | Description |
|---|---|---|
| PROCESS_ID | VARCHAR2(24) | Operating system process ID of the master |

| Column | Datatype | Description |
|---|---|---|
| PROCESS_NAME | VARCHAR2(48) | Operating system name of the master |
| NUM_JOBS | NUMBER | Number of jobs started |
| JOB_LATENCY | NUMBER | Job start latency |
| NUM_COORDINATORS | NUMBER | Number of masters started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

## V$AQ_BMAP_NONDUR_SUBSCRIBERS

V$AQ_BMAP_NONDUR_SUBSCRIBERS can be used to get the available bit positions. The view is queried to get the free bit position during creation of a non-durable subscriber.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Queue ID |
| USED_POS | RAW(128) | Stream of bits to identify used and available bit positions |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

## V$AQ_CROSS_INSTANCE_JOBS

V$AQ_CROSS_INSTANCE_JOBS describes each of the cross process jobs. Each job serves to forward messages for a shard from a source instance to a destination instance for a set of subscribers of a sharded queue.

| Column | Datatype | Description |
|---|---|---|
| JOB_ID | NUMBER | Job ID within this coordinator |

| Column | Datatype | Description |
|---|---|---|
| SCHEMA_NAME | VARCHAR2(30) | Source schema of the job |
| QUEUE_NAME | VARCHAR2(30) | Source queue name of the cross job |
| SHARD_ID | NUMBER | Source shard ID |
| START_SUBSHARD_ID | NUMBER | Start subshard ID of the job |
| DESTINATION_INSTANCE_ID | NUMBER | Destination instance of the cross job |
| COORDINATOR_ID | NUMBER | Index of the coordinator serving the job |
| DEST_SERVER_PROCESS_ID | NUMBER | Process ID of the destination server |
| JOB_STATE | VARCHAR2(9) | State of the job:<br>• REQUESTED<br>• RUNNING<br>• STOPPED<br>• PAUSED<br>• CRASHED<br>• INACTIVE |
| FLOW_CONTROL | NUMBER | Indicates whether the job is flow controlled:<br>• 0 - The job is not flow controlled<br>• 1 - The job is flow controlled |
| MSGS_SENT | NUMBER | Messages sent during the job |
| BYTES_SENT | NUMBER | Bytes sent during the job |
| ACK_LATENCY | NUMBER | Latency for receiving ACK for the job |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_JOB_COORDINATOR

V$AQ_JOB_COORDINATOR lists performance statistics per coordinator, for every AQ coordinator controlled by the Oracle Database Advanced Queueing master coordinator.

| Column | Datatype | Description |
|---|---|---|
| COORDINATOR_ID | NUMBER | ID of the coordinator |
| PROCESS_ID | VARCHAR2(24) | Operating system process ID of the coordinator |
| PROCESS_NAME | VARCHAR2(48) | Operating system process name of the coordinator |
| JOB_NAME | VARCHAR2(32) | Name of the job handled |
| JOB_TYPE | NUMBER | Type of job handled |
| SERVER_COUNT | NUMBER | Number of servers active |
| MAX_SERVER_COUNT | NUMBER | Maximum server fanout achieved |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_MESSAGE_CACHE

V$AQ_MESSAGE_CACHE provides performance statistics of the message cache for sharded queues at the subshard level in the instance.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Queue ID |
| SHARD_ID | NUMBER | Shard ID |
| PRIORITY | NUMBER | Priority of the subshard for dequeue, range 0-9 |
| SUBSHARD_ID | NUMBER | Subshard ID in the shard |
| PARTITION_ID | NUMBER | Partition id for the particular subshard |
| MAX_MSGS | NUMBER | Maximum number of messages of subshard |
| ENQUEUED_MSGS | NUMBER | Number of messages enqueued for the subshard |
| BROWSED_MSGS | NUMBER | Number of messages browsed (dequeued in browse mode) |
| MSGS_MADE_EXPIRED | NUMBER | Number of messages made expired |
| MSGS_MADE_READY | NUMBER | Number of messages made ready |
| CHUNK_SIZE | NUMBER | The size of the memory chunk for storing messages |
| NUM_CHUNKS | NUMBER | Number of chunks for the subshard |
| NUM_FREE_CHUNKS | NUMBER | Number of free chunks for the subshard |
| USED_MEMORY_SIZE | NUMBER | Total estimated size of memory in use (in bytes) for the subshard |

| Column | Datatype | Description |
|---|---|---|
| STATE | VARCHAR2(13) | Subshard state. Possible values:<br>• CACHED (in memory)<br>• UNCACHED (on disk)<br>• UNCACHED_FREE<br>• CACHED_FREE<br>• UNKNOWN |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_NONDUR_REGISTRATIONS

V$AQ_NONDUR_REGISTRATIONS provides information about non-durable subscriptions.

| Column | Datatype | Description |
|---|---|---|
| REG_ID | NUMBER | Registration ID |
| SUBSCRIPTION | VARCHAR2(128) | Subscription name |
| LOCATION | VARCHAR2(256) | Location name |
| USER# | NUMBER | User ID |
| USER_CONTEXT | RAW(32) | Context the user provided |
| CONTEXT_SIZE | NUMBER | Size of the context |
| NAMESPACE | NUMBER | Subscription namespace |
| VERSION | NUMBER | Database version number |
| STATE | NUMBER | State of the registration |
| QOS | NUMBER | Quality of service |
| REG_TIME | TIMESTAMP(3) WITH TIME ZONE | Time of registration |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_NONDUR_SUBSCRIBER

V$AQ_NONDUR_SUBSCRIBER provides information about the non-durable subscriptions on sharded queues.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Queue ID |
| SUBSCRIBER_ID | NUMBER | ID of the non-durable subscriber |
| SUBSCRIBER_NAME | VARCHAR2(30) | Subscriber name |
| RULE_CONDITION | VARCHAR2(4000) | Rule condition of the subscriber |
| TRANSFORMATION_OWNER | VARCHAR2(30) | Owner of the transformation (for JMS queues) |
| TRANSFORMATION_NAME | VARCHAR2(30) | Name of the transformation (for JMS queues) |
| CREATION_TIME | TIMESTAMP(1) WITH TIME ZONE | Non-durable subscriber creation time |
| FLAGS | NUMBER | Property of the subscriber:<br>• 0x0001 – Persistent subscriber<br>• 0x0002 – Buffered subscriber<br>• 0x0004 – JMS subscriber<br>• 0x0008 – Rule-based subscriber<br>• 0x0010 – Subscriber has transformation<br>• 0x0020 – Notification-only non-durable subscriber subscriber |
| SUBSCRIBER_TYPE | NUMBER | Type of subscriber:<br>• 1 – JMS non-durable subscriber<br>• 2 – Service layer dummy non-durable subscriber<br>• 3 – Notification-only non-durable subscriber |
| BITPOS | NUMBER | Position of subscriber in subscriber bitmap |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_NONDUR_SUBSCRIBER_LWM

V$AQ_NONDUR_SUBSCRIBER_LWM projects the low watermarks (LWMs) of non-durable subscribers in a sharded queue. The LWM of a non-durable subscriber is a combination of shard, priority and LWM (subshard).

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Queue ID |
| SUBSCRIBER_ID | NUMBER | ID of the non-durable subscriber |
| SHARD_ID | NUMBER | Shard ID |
| PRIORITY | NUMBER | Priority of the shard |
| LWM | NUMBER | Lower watermark (in a subshard) of the non-durable subscriber |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_NOTIFICATION_CLIENTS

V$AQ_NOTIFICATION_CLIENTS displays performance statistics for secure OCI client connections.

| Column | Datatype | Description |
|---|---|---|
| CLIENT_ID | VARCHAR2(29) | Internally generated client ID for secure notification clients |
| EMON_ID | NUMBER | Emon ID serving the client |
| NOTIFICATION_STATE | NUMBER | Notification state:<br>• ACTIVE – Sending notification<br>• WAIT_FOR_ACK – Waiting for client acknowledgment<br>• INACTIVE - Idle connection |
| NUM_MESSAGE_SENT | NUMBER | Number of messages sent on the connection |
| NUM_BYTES_SENT | NUMBER | Number of bytes sent on the connection |
| NUM_MESSAGE_RECEIVED | NUMBER | Number of messages successfully received by the client |
| LAST_SEND_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the last message was sent on the connection |
| LAST_RECEIVE_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the last message was received over the connection |
| CONNECT_TIME | TIMESTAMP(3) WITH TIME ZONE | Time at which the client connected |
| DISCONNECT_TIME | TIMESTAMP(3) WITH TIME ZONE | Time at which the client disconnected |
| LAST_ERROR | NUMBER | The last error that occurred on the client connection |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_SERVER_POOL

V$AQ_SERVER_POOL lists performance statistics for all the servers in the pool.

| Column | Datatype | Description |
|---|---|---|
| COORDINATOR_ID | NUMBER | ID of the coordinator |

| Column | Datatype | Description |
|---|---|---|
| COORDINATOR_INSTANCE_ID | NUMBER | Instance of the coordinator |
| PROCESS_ID | VARCHAR2(24) | Operating system process ID of the server |
| PROCESS_NAME | VARCHAR2(48) | Operating system process name of the server |
| JOB_NAME | VARCHAR2(32) | Name of the job handled |
| POOL_STATE | VARCHAR2(20) | State of the pool:<br>• REMOTE<br>• IDLE<br>• ACTIVE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$AQ_SUBSCRIBER_LOAD

V$AQ_SUBSCRIBER_LOAD describes the load of all subscribers of sharded queues in terms of latency at every instance in an Oracle RAC environment. Latency denotes the predicted amount of time (in seconds) required from the current time to drain all the messages for that subscriber at each respective instance. The latency calculation considers past enqueue/dequeue rates and future enqueue/dequeue rates based on history.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Queue ID |
| QUEUE_SCHEMA | VARCHAR2(30) | Queue schema |
| QUEUE_NAME | VARCHAR2(30) | Queue name |
| SUBSCRIBER_ID | NUMBER | Subscriber ID |
| SUBSCRIBER_NAME | VARCHAR2(30) | Subscriber name |
| LATENCY_STATE | VARCHAR2(8) | Possible values:<br>• FINITE - The subscriber will be able to dequeue all the messages in a finite amount of time<br>• INFINITE - The subscriber's dequeue rate may not catch up to the enqueue rate<br>• UNKNOWN - Latency is not yet known |
| LATENCY | NUMBER | Latency (in seconds). Valid only when LATENCY_STATE is FINITE. |
| DEQUEUE_REQUESTS | NUMBER | Approximate number of dequeue requests noted recently |

| Column | Datatype | Description |
|---|---|---|
| ACTIVE_SHARDS | NUMBER | Number of queue shards that have messages for this subscriber |
| ACTIVE_LISTENER | VARCHAR2(5) | Indicates whether the subscriber is actively listening at this instance for messages (TRUE) or not (FALSE) |
| FLAGS | NUMBER | For internal use only |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Queuing User's Guide* for more information about Oracle Database Advanced Queueing

# V$ARCHIVE

V$ARCHIVE displays information about redo log files in need of archiving. Each row displays information for one thread. This information is also available in V$LOG. Oracle recommends that you use V$LOG.

**See Also:**

"V$LOG"

| Column | Datatype | Description |
|---|---|---|
| GROUP# | NUMBER | Log file group number |
| THREAD# | NUMBER | Log file thread number |
| SEQUENCE# | NUMBER | Log file sequence number |
| ISCURRENT | VARCHAR2(3) | This is the current online redo log |
| CURRENT | VARCHAR2(3) | This column is obsolete and maintained for backward compatibility. The value of this column is always equal to the value in ISCURRENT. |
| FIRST_CHANGE# | NUMBER | First SCN stored in the current log |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ARCHIVE_DEST

V$ARCHIVE_DEST displays, for the current instance, all of the destinations in the Data Guard configuration, including each destination's current value, mode, and status.

| Column | Datatype | Description |
|---|---|---|
| DEST_ID | NUMBER | Log archive destination parameter identifier (1 to 31) |
| DEST_NAME | VARCHAR2(256) | Log archive destination parameter name |
| STATUS | VARCHAR2(9) | Identifies the current status of the destination:<br>• VALID - Initialized and available<br>• INACTIVE - No destination information<br>• DEFERRED - Manually disabled by the user<br>• ERROR - Error during open or copy<br>• DISABLED - Disabled after error<br>• BAD PARAM - Parameter has errors<br>• ALTERNATE - Destination is in an alternate state<br>• FULL - Exceeded quota size for the destination |
| BINDING | VARCHAR2(9) | Specifies how failure will affect the archival operation:<br>• MANDATORY - Successful archival is required<br>• OPTIONAL - Successful archival is not required (depends on LOG_ARCHIVE_MIN_SUCCEED_DEST) |
| NAME_SPACE | VARCHAR2(7) | Identifies the scope of parameter setting:<br>• SYSTEM - System definition<br>• SESSION - Session definition |
| TARGET | VARCHAR2(7) | Specifies whether the archive destination is local or remote to the primary database:<br>• PRIMARY - local<br>• STANDBY - remote |
| ARCHIVER | VARCHAR2(10) | Identifies the archiver process relative to the database where the query is issued:<br>• ARC*n*<br>• FOREGROUND<br>• LGWR<br>• RFS |
| SCHEDULE | VARCHAR2(8) | Indicates whether the archival of this destination is INACTIVE, PENDING, ACTIVE, or LATENT |
| DESTINATION | VARCHAR2(256) | Specifies the location where the archived redo logs are to be archived |
| LOG_SEQUENCE | NUMBER | Identifies the sequence number of the last archived redo log to be archived |
| REOPEN_SECS | NUMBER | Identifies the retry time (in seconds) after error |

| Column | Datatype | Description |
|---|---|---|
| DELAY_MINS | NUMBER | Identifies the delay interval (in minutes) before the archived redo log is automatically applied to a standby database |
| MAX_CONNECTIONS | NUMBER | Maximum number of connections |
| NET_TIMEOUT | NUMBER | Number of seconds the log writer process will wait for status from the network server of a network operation issued by the log writer process |
| PROCESS | VARCHAR2(10) | Identifies the archiver process relative to the primary database, even if the query is issued on the standby database:<br>• ARC*n*<br>• FOREGROUND<br>• LGWR |
| REGISTER | VARCHAR2(3) | Indicates whether the archived redo log is registered in the remote destination control file (YES) or not (NO). If the archived redo log is registered, it is available to log apply services. |
| FAIL_DATE | DATE | Date and time of last error |
| FAIL_SEQUENCE | NUMBER | Sequence number of the archived redo log being archived when the last error occurred |
| FAIL_BLOCK | NUMBER | Block number of the archived redo log being archived when the last error occurred |
| FAILURE_COUNT | NUMBER | Current number of contiguous archival operation failures that have occurred for the destination |
| MAX_FAILURE | NUMBER | Allows you to control the number of times log transport services will attempt to reestablish communication and resume archival operations with a failed destination |
| ERROR | VARCHAR2(256) | Displays the error text |
| ALTERNATE | VARCHAR2(256) | Alternate destination, if any |
| DEPENDENCY | VARCHAR2(256) | Reserved for future use |
| REMOTE_TEMPLATE | VARCHAR2(256) | Specifies the template to be used to derive the location to be recorded |
| QUOTA_SIZE | NUMBER | Destination quotas, expressed in bytes |
| QUOTA_USED | NUMBER | Size of all the archived redo logs currently residing on the specified destination |
| MOUNTID | NUMBER | Instance mount identifier |
| TRANSMIT_MODE | VARCHAR2(12) | Specifies network transmission mode:<br>• SYNCHRONOUS<br>• PARALLELSYNC<br>• ASYNCHRONOUS |
| ASYNC_BLOCKS | NUMBER | Number of blocks specified for the ASYNC attribute |
| AFFIRM | VARCHAR2(3) | Specifies disk I/O mode |

| Column | Datatype | Description |
|---|---|---|
| TYPE | VARCHAR2(7) | Indicates whether the archived log destination definition is PUBLIC or PRIVATE. Only PUBLIC destinations can be modified at run time using the ALTER SYSTEM SET or ALTER SESSION SET statements. By default, all archived log destinations are PUBLIC. |
| VALID_NOW | VARCHAR2(16) | Indicates whether the destination is valid right now for archival operations:<br>• YES - Redo log type and database role for this destination are valid for the current database<br>• WRONG VALID_TYPE - Redo log type specified for this destination is not valid for the current database role. For example, WRONG VALID_TYPE would be returned if a destination specified with the VALID_FOR=(STANDBY_LOGFILE,STANDBY_ROLE) attribute is running in the standby database role but does not have standby redo logs implemented.<br>• WRONG VALID_ROLE - Database role specified for this destination is not the role in which the database is currently running. For example, the WRONG VALID_ROLE would be returned when a destination defined with the VALID_FOR=(ONLINE_LOGFILE,STANDBY_ROLE) attribute is running in the primary database role.<br>• INACTIVE - Destination is inactive, probably due to an error |
| VALID_TYPE | VARCHAR2(15) | Redo log type or types that are valid for the destination:<br>• ONLINE_LOGFILE<br>• STANDBY_LOGFILE<br>• ALL_LOGFILES |
| VALID_ROLE | VARCHAR2(12) | Database role or roles that are valid for the destination:<br>• PRIMARY_ROLE<br>• STANDBY_ROLE<br>• ALL_ROLES |
| DB_UNIQUE_NAME | VARCHAR2(30) | Unique database name |
| VERIFY | VARCHAR2(3) | Indicates whether the value of the VERIFY attribute on the LOG_ARCHIVE_DEST_*n* parameter is verified (YES) or not verified (NO) |
| COMPRESSION | VARCHAR2(7) | Indicates whether network compression is ENABLED or DISABLED. |
| APPLIED_SCN | NUMBER | For a destination that corresponds to a physical or logical standby database, the SCN of the last applied redo. For a destination that corresponds to a snapshot standby database, the SCN of the last redo applied before conversion to a snapshot standby database. This column is only valid for enabled and active standby database destinations on a primary or cascading standby database. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |
| ENCRYPTION[1] | VARCHAR2(7) | Indicates whether encryption of the redo stream sent to Zero Data Loss Recovery Appliance (Recovery Appliance) is enabled (ENABLED) or not (DISABLED) |

1 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- *Zero Data Loss Recovery Appliance Administrator's Guide* for introductory information about Recovery Appliance
- "LOG_ARCHIVE_DEST" and "LOG_ARCHIVE_DEST_n"
- "LOG_ARCHIVE_DUPLEX_DEST" and "LOG_ARCHIVE_DEST_STATE_n"
- "STANDBY_ARCHIVE_DEST"
- "LOG_ARCHIVE_MIN_SUCCEED_DEST"

# V$ARCHIVE_DEST_STATUS

V$ARCHIVE_DEST_STATUS displays run time and configuration information for the archived redo log destinations. The information in this view does not persist across an instance shutdown.

| Column | Datatype | Description |
|---|---|---|
| DEST_ID | NUMBER | Identifies the log archive destination parameter (1 to 31) |
| DEST_NAME | VARCHAR2(256) | Log archive destination parameter name |
| STATUS | VARCHAR2(9) | Current status of the destination:<br>• VALID - Initialized and available<br>• INACTIVE - No destination information<br>• DEFERRED - Manually disabled by the user<br>• ERROR - Error during open or copy<br>• DISABLED - Disabled after error<br>• BAD PARAM - Parameter has errors<br>• ALTERNATE - Destination is in an alternate state<br>• FULL - Exceeded quota size for the destination |
| TYPE | VARCHAR2(14) | Type of archival destination database:<br>• LOCAL - Local to primary database<br>• PHYSICAL - Physical standby<br>• CROSS-INSTANCE - An instance of the primary<br>• LOGICAL - Logical standby<br>• SNAPSHOT - Snapshot standby database<br>• DOWNSTREAM - Streams downstream capture database<br>• FAR SYNC - Far Sync Instance |

| Column | Datatype | Description |
|---|---|---|
| DATABASE_MODE | VARCHAR2(15) | Current mode of the archival destination database:<br>• STARTED - Instance started, not mounted<br>• MOUNTED - Mounted<br>• MOUNTED-STANDBY - Mounted standby<br>• OPEN - Open read/write<br>• OPEN_READ-ONLY - Open read-only |
| RECOVERY_MODE | VARCHAR2(23) | Current apply mode at the archival destination:<br>• IDLE - Managed recovery is not active<br>• MANAGED - Managed recovery is active<br>• MANAGED REAL TIME APPLY - Log apply services recover redo data from standby redo logs at the same time the logs are being written to, as opposed to recovering redo from archived redo logs when a log switch occurs<br>• LOGICAL REAL TIME APPLY - Real time SQL Apply<br>• LOGICAL APPLY - SQL Apply |
| PROTECTION_MODE | VARCHAR2(20) | Indicates whether the database is protected:<br>• MAXIMUM PROTECTION<br>• MAXIMUM AVAILABILITY<br>• RESYNCHRONIZATION<br>• MAXIMUM PERFORMANCE<br>• UNPROTECTED |
| DESTINATION | VARCHAR2(256) | Specifies the location where the redo data is to be archived |
| STANDBY_LOGFILE_COUNT | NUMBER | Indicates the total number of standby redo logs created on the standby database |
| STANDBY_LOGFILE_ACTIVE | NUMBER | Indicates the total number of standby redo logs on the standby database that are active and contain primary database online redo log information |
| ARCHIVED_THREAD# | NUMBER | Identifies the thread number of the most recent archived redo log received at the destination |
| ARCHIVED_SEQ# | NUMBER | Identifies the log sequence number of the most recent archived redo log received at the destination |
| APPLIED_THREAD# | NUMBER | Identifies the thread number of the most recent applied redo log received at the destination |
| APPLIED_SEQ# | NUMBER | Identifies the log sequence number of the most recent applied redo log received at the destination |
| ERROR | VARCHAR2(256) | Displays the error text |
| SRL | VARCHAR2(3) | Indicates whether standby redo logfiles are used on the standby database (YES) or not (NO) |
| DB_UNIQUE_NAME | VARCHAR2(30) | Specifies the unique database name of the current instance that was defined with the DB_UNIQUE_NAME attribute on the LOG_ARCHIVE_DEST_*n* parameter |
| SYNCHRONIZATION_STATUS | VARCHAR2(22) | Possible values for this column are as follows:<br>• CHECK CONFIGURATION - Synchronization with this destination is not possible because this database is either not in MAXIMUM PROTECTION or MAXIMUM PERFORMANCE data |

| Column | Datatype | Description |
|---|---|---|
| | | protection mode, or the LOG_ARCHIVE_DEST_*n* parameter associated with this destination has not been configured with the SYNC and AFFIRM attributes.<br>• CHECK STANDBY REDO LOG - The standby redo log at this destination is configured improperly.<br>• CHECK NETWORK - One or more instances of this database cannot send redo data to this destination.<br>• DESTINATION HAS A GAP - This destination is missing redo data needed for synchronization with this database.<br>• OK - This destination is synchronized with this database.<br>• NOT AVAILABLE - Synchronization status is not available.<br>**See Also:** *Oracle Data Guard Concepts and Administration* for more information about redo transport configuration |
| SYNCHRONIZED | VARCHAR2(3) | Possible values are:<br>• YES - This destination is synchronized with the primary database.<br>• NO - The destination is not synchronized with the primary database.<br>• UNKNOWN - The synchronization status of this destination cannot be determined. |
| GAP_STATUS | VARCHAR2(24) | Redo gap status:<br>• NO GAP - Destination does not have a redo gap.<br>• LOG SWITCH GAP - Destination has not yet received all of the redo from the previous log file.<br>• RESOLVABLE GAP - Destination has a redo gap that can be automatically resolved by fetching the missing redo from this database.<br>• UNRESOLVABLE GAP - Destination has a redo gap that cannot be automatically resolved by fetching the missing redo from this database and there are no other destinations from which redo can be fetched.<br>• LOCALLY UNRESOLVABLE GAP - Destination has a redo gap that cannot be automatically resolved by fetching the missing redo from this database. It may be possible to resolve the gap by fetching the missing redo from another destination. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ARCHIVE_GAP

V$ARCHIVE_GAP displays information about archive gaps on a standby database. This view can be used to find out the current archive gap that is blocking recovery for the current recovery incarnation.

| Column | Datatype | Description |
|---|---|---|
| THREAD# | NUMBER | Thread number of the missing archived redo log files. The number is 1 for a single instance. For Oracle Real Application Clusters, this column will contain different numbers. |
| LOW_SEQUENCE# | NUMBER | Lowest sequence number of the log files received on the standby system |
| HIGH_SEQUENCE# | NUMBER | Highest sequence number of the log files received on the standby system |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ARCHIVE_PROCESSES

V$ARCHIVE_PROCESSES displays the state of the various ARCH processes for the instance.

| Column | Datatype | Description |
|---|---|---|
| PROCESS | NUMBER | Identifier for the ARCH process for the instance, numbered from 0-9 |
| STATUS | VARCHAR2(10) | Status of the ARCH process, displayed as a keyword. Possible values are: STOPPED, SCHEDULED, STARTING, ACTIVE, STOPPING, and TERMINATED. |
| LOG_SEQUENCE | NUMBER | This is the online redo log sequence number currently being archived, if STATE="BUSY" |
| STATE | VARCHAR2(4) | This is the current state of the ARCH process, displayed as a keyword. Possible keywords are IDLE or BUSY. |
| ROLES | VARCHAR2(36) | The list of roles assigned to the archive process. The roles include: HEART_BEAT, NO_FAL, NO_SRL, CLEAR_LOGS |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ARCHIVED_LOG

V$ARCHIVED_LOG displays archived log information from the control file, including archive log names. An archive log record is inserted after the online redo log is successfully archived or cleared (name column is NULL if the log was cleared). If the

log is archived twice, there will be two archived log records with the same THREAD#, SEQUENCE#, and FIRST_CHANGE#, but with a different name. An archive log record is also inserted when an archive log is restored from a backup set or a copy and whenever a copy of a log is made with the RMAN COPY command.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Archived log record ID |
| STAMP | NUMBER | Archived log record stamp |
| NAME | VARCHAR2(513) | Archived log file name. If set to NULL, either the log file was cleared before it was archived or an RMAN backup command with the "delete input" option was executed to back up archivelog all (RMAN> backup archivelog all delete input;). |
| DEST_ID | NUMBER | Original destination from which the archive log was generated. The value is 0 if the destination identifier is not available. |
| THREAD# | NUMBER | Redo thread number |
| SEQUENCE# | NUMBER | Redo log sequence number |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change number of the database when the log was written |
| RESETLOGS_TIME | DATE | Resetlogs time of the database when the log was written |
| RESETLOGS_ID | NUMBER | Resetlogs identifier associated with the archived redo log |
| FIRST_CHANGE# | NUMBER | First change number in the archived log |
| FIRST_TIME | DATE | Timestamp of the first change |
| NEXT_CHANGE# | NUMBER | First change in the next log |
| NEXT_TIME | DATE | Timestamp of the next change |
| BLOCKS | NUMBER | Size of the archived log (in blocks) |
| BLOCK_SIZE | NUMBER | Redo log block size. This is the logical block size of the archived log, which is the same as the logical block size of the online log from which the archived log was copied. The online log logical block size is a platform-specific value that is not adjustable by the user. |
| CREATOR | VARCHAR2(7) | Creator of the archivelog:<br>• ARCH - Archiver process<br>• FGRD - Foreground process<br>• RMAN - Recovery Manager<br>• SRMN - RMAN at standby<br>• LGWR - Logwriter process |
| REGISTRAR | VARCHAR2(7) | Registrar of the entry:<br>• RFS - Remote File Server process<br>• ARCH - Archiver process<br>• FGRD - Foreground process<br>• RMAN - Recovery manager<br>• SRMN - RMAN at standby |

| Column | Datatype | Description |
|---|---|---|
| | | • LGWR - Logwriter process |
| STANDBY_DEST | VARCHAR2(3) | Indicates whether the entry is an archivelog destination (YES) or not (NO) |
| ARCHIVED | VARCHAR2(3) | Indicates whether the online redo log was archived (YES) or whether RMAN only inspected the log and created a record for future application of redo logs during recovery (NO). |
| APPLIED | VARCHAR2(9) | Indicates whether an archived redo log file has been applied to the corresponding physical standby database. The value is always NO for local destinations.<br>This column is meaningful on a physical standby database for rows where REGISTRAR = RFS:<br>• If REGISTRAR = RFS and APPLIED = NO, then the log file has been received but has not yet been applied.<br>• If REGISTRAR = RFS and APPLIED = IN-MEMORY, then the log file has been applied in memory, but the data files have not yet been updated.<br>• If REGISTRAR = RFS and APPLIED = YES, then the log file has been applied and the data files have been updated.<br>This column can be used to identify log files that can be backed up and deleted. When used for this purpose, the value IN-MEMORY should be treated as if it were NO. |
| DELETED | VARCHAR2(3) | Indicates whether an RMAN DELETE command has physically deleted the archived log file from disk, as well as logically removing it from the control file of the target database and from the recovery catalog (YES) or not (NO) |
| STATUS | VARCHAR2(1) | Status of the archived log:<br>A - Available<br>D - Deleted<br>U - Unavailable<br>X - Expired |
| COMPLETION_TIME | DATE | Time when the archiving completed |
| DICTIONARY_BEGIN | VARCHAR2(3) | Indicates whether the log contains the start of a LogMiner dictionary (YES) or not (NO) |
| DICTIONARY_END | VARCHAR2(3) | Indicates whether the log contains the end of a LogMiner dictionary (YES) or not (NO) |
| END_OF_REDO | VARCHAR2(3) | Indicates whether the archived redo log contains the end of all redo information from the primary database (YES) or not (NO) |
| BACKUP_COUNT | NUMBER | Indicates the number of times this file has been backed up. Values range from 0-15. If the file has been backed up more than 15 times, the value remains 15. |
| ARCHIVAL_THREAD# | NUMBER | Redo thread number of the instance that performed the archival operation. This column differs from the THREAD# column only when a closed thread is archived by another instance. |
| ACTIVATION# | NUMBER | Number assigned to the database instantiation |

| Column | Datatype | Description |
|---|---|---|
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether the file was created in the fast recovery area (YES) or not (NO) |
| COMPRESSED | VARCHAR2(3) | Reserved for internal use |
| FAL | VARCHAR2(3) | Indicates whether the archive log was generated as the result of a FAL request (YES) or not (NO) |
| END_OF_REDO_TYPE | VARCHAR2(10) | Possible values are as follows:<br>• SWITCHOVER - Shows archived redo log files that are produced at the end of a switchover<br>• TERMINAL - Shows archived redo log files produced after a failover<br>• RESETLOGS - Shows online redo log files archived on the primary database after an ALTER DATABASE OPEN RESETLOGS statement is issued<br>• ACTIVATION - Shows any log files archived on a physical standby database after an ALTER DATABASE ACTIVATE STANDBY DATABASE statement is issued<br>• "empty string" - Any empty string implies that the log is just a normal archival and was not archived due to any of the other events |
| BACKED_BY_VSS | VARCHAR2(3) | Whether or not the file has been backed up by Volume Shadow Copy Service (VSS). This column is reserved for internal use. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ASM_ACFS_ENCRYPTION_INFO

V$ASM_ACFS_ENCRYPTION_INFO displays encryption information for every mounted Oracle ACFS.

| Column | Datatype | Description |
|---|---|---|
| FS_NAME | VARCHAR2(1024) | File system mount point (foreign key to V$ASM_FILESYSTEM) |
| VOL_DEVICE | VARCHAR2(256) | Name of the Oracle ADVM device |
| SET_STATUS | VARCHAR2(7) | Indicates whether encryption parameters have been set on the file system (YES) or not (NO); otherwise UNKNOWN. |
| ENABLED_STATUS | VARCHAR2(8) | Indicates whether file system level encryption is enabled (ENABLED) or not (DISABLED); otherwise UNKNOWN. |
| ALGORITHM | VARCHAR2(7) | Encryption algorithm used. AES is the only supported algorithm. |
| KEY_LENGTH | VARCHAR2(7) | Key length used for the encryption key |

| Column | Datatype | Description |
|---|---|---|
| LAST_REKEY_TIME | DATE | Time that the volume was last rekeyed |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_ADMIN

V$ASM_ACFS_SEC_ADMIN contains Oracle ACFS security administrators in the cluster. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| ADMIN_NAME | VARCHAR2(32) | User name of the security administrator |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_CMDRULE

V$ASM_ACFS_SEC_CMDRULE contains one row for every Oracle ACFS security command rule for each Oracle ACFS file system. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| CMD_RULE_NAME | VARCHAR2(32) | Name of the command rule |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_REALM

V$ASM_ACFS_SEC_REALM contains an entry for every realm in the Oracle ACFS security file system in an Oracle Automatic Storage Management (Oracle ASM) instance. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| REALM_NAME | VARCHAR2(255) | Realm name |
| REALM_STATUS | NUMBER | Realm status:<br>• 0: disable<br>• 1: enable |
| REALM_ENCR_STATUS | NUMBER | Realm encryption status:<br>• 0: disable<br>• 1: enable<br>Will be provided only when realm encryption status is 1. The values will be among 128, 192, 256. |
| REALM_ENCR_KEYLEN | NUMBER | Encryption key length.<br>Will be provided only when realm encryption status is 1. The values will be among 128, 192, 256. |
| REALM_ENCR_ALGO | VARCHAR2(7) | Encryption algorithm.<br>Will be provided only when realm encryption status is 1. |
| REALM_DESC | VARCHAR2(1024) | Realm description |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_REALM_FILTER

V$ASM_ACFS_SEC_REALM_FILTER contains an entry for every filter in the Oracle ACFS security realm for each Oracle ACFS file system. A filter is a defined as a command rule/rule set pair in a realm. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| REALM_NAME | VARCHAR2(255) | Realm name |
| CMD_RULE_NAME | VARCHAR2(32) | Command rule name |
| RULESET_NAME | VARCHAR2(256) | Ruleset name |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_REALM_GROUP

V$ASM_ACFS_SEC_REALM_GROUP contains an entry for every group in the Oracle ACFS security realm for each Oracle ACFS file system. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| REALM_NAME | VARCHAR2(255) | Realm name |
| GROUP_NAME | VARCHAR2(256) | Name of the group |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_REALM_USER

V$ASM_ACFS_SEC_REALM_USER contains an entry for every user in the Oracle ACFS security realm for each Oracle ACFS file system. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| REALM_NAME | VARCHAR2(255) | Realm name |
| USER_NAME | VARCHAR2(32) | Name of the user |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_RULE

V$ASM_ACFS_SEC_RULE contains information about all Oracle ACFS security rules for each Oracle ACFS file system. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| RULE_NAME | VARCHAR2(256) | Name of the rule |
| RULE_TYPE | VARCHAR2(32) | Type of the rule:<br>• time<br>• username<br>• groupname<br>• hostname<br>• application |
| RULE_VALUE | VARCHAR2(512) | Value of the rule, which could include values such as 23:10:00 for time, Bob for username, dba for groupname, host147 for hostname, /sbin/cat for application. |
| RULE_VALUE2 | VARCHAR2(512) | The value will be populated only if the rule type is time. When the rule type is time, the start time is entered in RULE_VALUE and the end time is entered in RULE_VALUE2. |
| RULE_OPTION | VARCHAR2(7) | Possible values are ALLOW or DENY |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_RULESET

V$ASM_ACFS_SEC_RULESET contains information about all Oracle ACFS security rule sets for each Oracle ACFS file system. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| RULESET_NAME | VARCHAR2(256) | Name of the ruleset |
| RULESET_OPTION | VARCHAR2(8) | The possible values are ANY_TRUE or ALL_TRUE |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SEC_RULESET_RULE

V$ASM_ACFS_SEC_RULESET_RULE contains an entry for every rule in the Oracle ACFS security rule set for each Oracle ACFS file system. This view also supports GV$ global views.

| Column | Datatype | Description |
|---|---|---|
| RULESET_NAME | VARCHAR2(256) | Rule set name |
| RULE_NAME | VARCHAR2(256) | Rule name |
| FS_NAME | VARCHAR2(1024) | Mount path name to identify the file system |
| VOL_DEVICE | VARCHAR2(256) | ADVM volume name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFS_SECURITY_INFO

V$ASM_ACFS_SECURITY_INFO displays security information for every mounted Oracle ACFS.

| Column | Datatype | Description |
|---|---|---|
| FS_NAME | VARCHAR2(1024) | File system mount point (foreign key to V$ASM_FILESYSTEM) |
| VOL_DEVICE | VARCHAR2(256) | Name of the Oracle ADVM device |
| PREPARED_STATUS | VARCHAR2(7) | Indicates whether the file system is prepared for security (YES) or not (NO); otherwise UNKNOWN. |
| ENABLED_STATUS | VARCHAR2(8) | Indicates whether security is enabled for the file system (ENABLED) or not (DISABLED); otherwise UNKNOWN. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFSREPL

V$ASM_ACFSREPL displays information for Oracle ACFS file systems that are initialized for replication.

| Column | Datatype | Description |
|---|---|---|
| FSNAME | VARCHAR2(1024) | File system mount point (foreign key to V$ASM_FILESYSTEM) |
| VOLDEV | VARCHAR2(256) | Name of the Oracle ADVM device |
| SITE | VARCHAR2(7) | Replication site role:<br>• PRIMARY - File system is initialized as a primary<br>• STANDBY - File system is initialized as a standby |
| LAG | VARCHAR2(128) | Amount of time standby is behind primary in hours:minutes:seconds (primary only) |
| STATUS | VARCHAR2(12) | Replication primary or standby status:<br>• ONLINE - Replication is initialized and processing real-time changes<br>• INITIALIZING - Replication is still processing files on the primary that existed before replication started<br>• PAUSED - The primary is not sending logs to the standby or the standby is not applying logged changes |
| INITSTATUS | VARCHAR2(11) | Replication primary or standby initialization status:<br>• NONE - Replication is ONLINE and not initializing<br>• DIRECTORIES - Directories are being initialized<br>• FILES - Files are being initialized |
| DIRSSCAN | NUMBER | Number of directories scanned and logged (primary only) |
| PERCENT | NUMBER | Percentage of files initialized that existed before replication started (primary only) |

| Column | Datatype | Description |
|---|---|---|
| LASTSYNC | DATE | Time of last apply on the standby site (standby only) |
| CRSRUN | NUMBER | Number of Cluster Ready Services replication daemons currently running |
| CRSTOTAL | NUMBER | Number of Cluster Ready Services replication daemons expected to be running |
| PMNTPT | VARCHAR2(1024) | Replication primary site mount point |
| SMNTPT | VARCHAR2(1024) | Replication standby site mount point |
| PSVCNAME | VARCHAR2(1024) | Replication primary site service name |
| SSVCNAME | VARCHAR2(1024) | Replication standby site service name |
| PHOST | VARCHAR2(1024) | A replication primary cluster host name |
| SHOST | VARCHAR2(1024) | A replication standby cluster host name |
| REMALIAS | VARCHAR2(4096) | Replication remote alias name |
| TAGS | NUMBER | TRUE (1) if replicating tagged files |
| COMPRESSION | VARCHAR2(3) | Replication log compression status:<br>• OFF - Replication logs are not being compressed<br>• ON - Replication logs are being compressed |
| DBGLVL | NUMBER | Replication trace log debug level |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFSREPLTAG

V$ASM_ACFSREPLTAG displays replicated tag information for Oracle ACFS file systems that are initialized for replication.

| Column | Datatype | Description |
|---|---|---|
| FSNAME | VARCHAR2(1024) | File system mount point (foreign key to V$ASM_FILESYSTEM) |
| VOLDEV | VARCHAR2(256) | Name of the Oracle ADVM device |
| TAG | VARCHAR2(32) | Tag name that is being replicated |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFSSNAPSHOTS

V$ASM_ACFSSNAPSHOTS displays snapshot information for every mounted Oracle ACFS.

| Column | Datatype | Description |
|---|---|---|
| FS_NAME | VARCHAR2(1024) | File system mount point (foreign key to V$ASM_FILESYSTEM) |
| VOL_DEVICE | VARCHAR2(256) | Name of the Oracle ADVM device |
| SNAP_NAME | VARCHAR2(1024) | Name of the snapshot |
| CREATE_TIME | DATE | Time when the snapshot was created |
| TYPE | VARCHAR2(2) | Snapshot type. Possible types are read-only (RO) or read/write (RW). |
| PARENT | VARCHAR2(1024) | Parent name used to create the snapshot. If the snapshot was created using the mount point as a base, parent name will be NULL. If the snapshot was created using an existing snapshot as the base, the parent name will be the name of the snapshot passed to the `acfsutil snap create -p` operation. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFSTAG

V$ASM_ACFSTAG displays every file or directory that has a tag and its tag name for every mounted Oracle ACFS.

| Column | Datatype | Description |
|---|---|---|
| TAG_NAME | VARCHAR2(32) | Tag name on file or directory |
| FS_NAME | VARCHAR2(1024) | File system mount point |
| PATH_NAME | VARCHAR2(4096) | Complete file path name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ACFSVOLUMES

V$ASM_ACFSVOLUMES displays information about mounted Oracle ACFS volumes, correlated with V$ASM_FILESYSTEM.

| Column | Datatype | Description |
|---|---|---|
| FS_NAME | VARCHAR2(1024) | File system mount point (foreign key to V$ASM_FILESYSTEM) |
| VOL_DEVICE | VARCHAR2(256) | Name of the Oracle ADVM device |
| VOL_LABEL | VARCHAR2(64) | Volume label (optional name) assigned through mkfs/acfsformat; NULL if no name exists |
| PRIMARY_VOL | VARCHAR2(5) | Indicates whether the volume is the primary volume for the file system (TRUE) or not (FALSE) |
| TOTAL_MB | NUMBER | Size of the volume device (in megabytes) |
| FREE_MB | NUMBER | Available space on the volume device (in megabytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- "V$ASM_FILESYSTEM"
- *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM) information

**Note:**

To display information about Oracle ACFS file systems or volumes that are located on nodes in an Oracle Flex ASM configuration, you must connect to the Oracle ASM proxy instance instead of the local Oracle ASM instance. For information about Oracle Flex ASM, refer to *Oracle Automatic Storage Management Administrator's Guide*.

# V$ASM_ALIAS

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_ALIAS displays one row for every alias present in every disk group mounted by the Oracle ASM instance.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(70) | Oracle ASM alias or alias directory name |
| GROUP_NUMBER | NUMBER | Owning disk group number of the alias (foreign key to the V$ASM_DISKGROUP view) |
| FILE_NUMBER | NUMBER | Oracle ASM file number of the alias (foreign key to the V$ASM_FILE view) |
| FILE_INCARNATION | NUMBER | Oracle ASM file incarnation number for the alias |
| ALIAS_INDEX | NUMBER | Alias entry number for the alias |
| ALIAS_INCARNATION | NUMBER | Incarnation number for the parent of the alias |
| PARENT_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and an alias entry number in the low-order 24 bits (number of the directory containing the alias) |
| REFERENCE_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and an alias entry number in the low-order 24 bits (number of the directory describing the current entry) |
| ALIAS_DIRECTORY | VARCHAR2(1) | Indicates whether the alias is to a directory (Y) or to an Oracle ASM file (N) |
| SYSTEM_CREATED | VARCHAR2(1) | Indicates whether the alias is system created (Y) or user created (N) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_ATTRIBUTE

V$ASM_ATTRIBUTE displays one row for each attribute defined. In addition to attributes specified by CREATE DISKGROUP and ALTER DISKGROUP statements, the

view may show other attributes that are created automatically. Note that attributes are only displayed for disk groups where COMPATIBLE.ASM is set to 11.1 or higher.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(256) | Full name of the attribute |
| VALUE | VARCHAR2(256) | Value of the attribute |
| GROUP_NUMBER | NUMBER | Number of the disk group in which this attribute exists (composite primary key) |
| ATTRIBUTE_INDEX | NUMBER | Number of this attribute in the disk group (composite primary key) |
| ATTRIBUTE_INCARNATION | NUMBER | Incarnation number for this attribute (composite primary key) |
| READ_ONLY | VARCHAR2(7) | Indicates whether the attribute is read-only (Y) or not (N) |
| SYSTEM_CREATED | VARCHAR2(7) | Indicates whether this is a system-created attribute (Y) or not (N) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Automatic Storage Management Administrator's Guide* for more information about viewing Oracle ASM disk group attributes using this view
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_AUDIT_CLEAN_EVENTS

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_AUDIT_CLEAN_EVENTS displays information about the history of audit trail cleanup or purge events. In a database instance, V$ASM_AUDIT_CLEAN_EVENTS displays no rows.

| Column | Datatype | Description |
|---|---|---|
| AUDIT_TRAIL | VARCHAR2(64) | Audit trail that was cleaned at the time of the event: OS AUDIT TRAIL |
| CLEANUP_TIME | TIMESTAMP(6) WITH TIME ZONE | Timestamp when the cleanup event completed |

| Column | Datatype | Description |
|---|---|---|
| DELETE_COUNT | NUMBER | Number of audit files that were deleted at the time of the event |
| WAS_FORCED | VARCHAR2(64) | Indicates whether a forced cleanup occurred (YES) or not (NO); forced cleanup bypasses the last archive timestamp set |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_AUDIT_CLEANUP_JOBS

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_AUDIT_CLEANUP_JOBS displays information about the configured audit trail purge jobs. In a database instance, V$ASM_AUDIT_CLEANUP_JOBS displays no rows.

| Column | Datatype | Description |
|---|---|---|
| JOB_NAME | VARCHAR2(64) | Name of the audit trail purge job |
| JOB_STATUS | VARCHAR2(64) | Current status of the audit trail purge job (ENABLED) or (DISABLED) |
| AUDIT_TRAIL | VARCHAR2(64) | Audit trail for which the audit trail purge job is configured: OS AUDIT TRAIL and UNIFIED AUDIT TRAIL |
| JOB_FREQUENCY | VARCHAR2(64) | Frequency at which the audit trail purge job runs |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_AUDIT_CONFIG_PARAMS

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_AUDIT_CONFIG_PARAMS displays information about the currently configured audit trail properties that are used by the DBMS_AUDIT_MGMT package. In a database instance, V$ASM_AUDIT_CONFIG_PARAMS displays no rows.

| Column | Datatype | Description |
|---|---|---|
| PARAMETER_NAME | VARCHAR2(64) | Name of the property |
| PARAMETER_VALUE | VARCHAR2(64) | Value of the property |
| AUDIT_TRAIL | VARCHAR2(64) | Audit trails for which the property is configured: OS AUDIT TRAIL and UNIFIED AUDIT TRAIL |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_AUDIT_MGMT package
- *Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_AUDIT_LAST_ARCH_TS

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_AUDIT_LAST_ARCH_TS displays information about the last archive timestamps set for audit trail cleanup or purges. In a database instance, V$ASM_AUDIT_LAST_ARCH_TS displays no rows.

| Column | Datatype | Description |
|---|---|---|
| AUDIT_TRAIL | VARCHAR2(64) | Audit trail for which the last archive timestamp applies: OS AUDIT TRAIL and UNIFIED AUDIT TRAIL |
| LAST_ARCHIVE_TS | TIMESTAMP(6) WITH TIME ZONE | Timestamp of the last audit file that has been archived |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_CLIENT

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_CLIENT identifies databases using disk groups managed by the Oracle ASM instance. In a database instance, V$ASM_CLIENT displays information about the Oracle ASM instance if the database has any open Oracle ASM files.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Number of the disk group being used by the client database instance (foreign key to the V$ASM_DISKGROUP view). When a client is connected to an Oracle ASM instance, but is not currently using any disk groups, V$ASM_CLIENT.GROUP_NUMBER contains 0. |
| INSTANCE_NAME | VARCHAR2(64) | Identifier for the database instance client |
| DB_NAME | VARCHAR2(8) | Unique database name of the database client instance |
| CLUSTER_NAME | VARCHAR2(31) | Name of the cluster |
| STATUS | VARCHAR2(12) | Status of the client connection:<br>• CONNECTED - Database instance client has an active connection to the Oracle ASM instance<br>• DISCONNECTED - Database instance client normally ended its connection to the Oracle ASM instance<br>• BROKEN - Connection with the database instance client terminated abnormally |
| SOFTWARE_VERSION | VARCHAR2(60) | Software version number of the database or Oracle ASM instance for the selected disk group connection |
| COMPATIBLE_VERSION | VARCHAR2(60) | Compatibility setting of the database or Oracle ASM instance for the selected disk group connection |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_DISK

V$ASM_DISK displays one row for every disk discovered by the Oracle Automatic Storage Management (Oracle ASM) instance, including disks that are not part of any disk group.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Number of the disk group containing the disk (foreign key to the V$ASM_DISKGROUP view) |
| DISK_NUMBER | NUMBER | Number assigned to the disk within its disk group |
| COMPOUND_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and a disk number in the low-order 24 bits (for efficient access to the view) |
| INCARNATION | NUMBER | Incarnation number for the disk |
| MOUNT_STATUS | VARCHAR2(7) | Per-instance status of the disk relative to group mounts:<br>• MISSING - Oracle ASM metadata indicates that the disk is known to be part of the Oracle ASM disk group but no disk in the storage system was found with the indicated name<br>• CLOSED - Disk is present in the storage system but is not being accessed by Oracle ASM<br>• OPENED - Disk is present in the storage system and is being accessed by Oracle ASM. This is the normal state for disks in a database instance which are part of a disk group being actively used by the instance.<br>• CACHED - Disk is present in the storage system and is part of a disk group being accessed by the Oracle ASM instance. This is the normal state for disks in an Oracle ASM instance which are part of a mounted disk group.<br>• IGNORED - Disk is present in the system but is ignored by Oracle ASM because of one of the following:<br>– The disk is detected by the system library but is ignored because an Oracle ASM library discovered the same disk<br>– Oracle ASM has determined that the membership claimed by the disk header is no longer valid<br>• CLOSING - Oracle ASM is in the process of closing this disk |
| HEADER_STATUS | VARCHAR2(12) | Per-instance status of the disk as seen by discovery:<br>• UNKNOWN - Oracle ASM disk header has not been read<br>• CANDIDATE - Disk is not part of a disk group and may be added to a disk group with the ALTER DISKGROUP statement<br>• INCOMPATIBLE - Version number in the disk header is not compatible with the Oracle ASM software version<br>• PROVISIONED - Disk is not part of a disk group and may be added to a disk group with the ALTER DISKGROUP statement. The PROVISIONED header status is different from the CANDIDATE header status in that PROVISIONED implies |

| Column | Datatype | Description |
|---|---|---|
| | | that an additional platform-specific action has been taken by an administrator to make the disk available for Oracle ASM.<br>• MEMBER - Disk is a member of an existing disk group. No attempt should be made to add the disk to a different disk group. The ALTER DISKGROUP statement will reject such an addition unless overridden with the FORCE option.<br>• FORMER - Disk was once part of a disk group but has been dropped cleanly from the group. It may be added to a new disk group with the ALTER DISKGROUP statement.<br>• CONFLICT - Oracle ASM disk was not mounted due to a conflict<br>• FOREIGN - Disk contains data created by an Oracle product other than Oracle ASM. This includes data files, logfiles, and OCR disks. |
| MODE_STATUS | VARCHAR2(7) | Global status about which kinds of I/O requests are allowed to the disk:<br>• ONLINE - Disk is online and operating normally. Reads and writes are attempted when the disk is mounted. Reads are attempted as part of disk discovery.<br>• OFFLINE - Disk is offline and access to data is not permitted. Reads and writes are not attempted. An offline disk remains logically part of its disk group. |
| STATE | VARCHAR2(8) | Global state of the disk with respect to the disk group:<br>• UNKNOWN - Oracle ASM disk state is not known (typically the disk is not mounted)<br>• NORMAL - Disk is online and operating normally<br>• ADDING - Disk is being added to a disk group, and is pending validation by all instances that have the disk group mounted<br>• DROPPING - Disk has been manually taken offline and space allocation or data access for the disk halts. Rebalancing will commence to relocate data off the disks to other disks in the disk group. Upon completion of the rebalance, the disk is expelled from the group.<br>• HUNG - Disk drop operation cannot continue because there is insufficient space to relocate the data from the disk being dropped<br>• FORCING - Disk is being removed from the disk group without attempting to offload its data. The data will be recovered from redundant copies, where possible.<br>• DROPPED - Disk has been fully expelled from the disk group |
| REDUNDANCY | VARCHAR2(7) | Hardware redundancy of the disk:<br>• UNKNOWN<br>• UNPROT<br>• MIRROR<br>• PARITY<br>**Note:** This column is valid only if an ASMLIB is present that supports returning hardware redundancy information. This column is not related to the redundancy of the disk group of which the disk is a member. |
| LIBRARY | VARCHAR2(64) | Name of the library that discovered the disk |
| OS_MB | NUMBER | Size of the disk (in megabytes) as reported by the host operating system |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_MB | NUMBER | Total capacity of the disk (in megabytes) |
| FREE_MB | NUMBER | Unused capacity of the disk (in megabytes) |
| HOT_USED_MB | NUMBER | Number of used megabytes in the hot region |
| COLD_USED_MB | NUMBER | Number of used megabytes in the cold region |
| NAME | VARCHAR2(30) | Name of the disk |
| FAILGROUP | VARCHAR2(30) | Name of the failure group containing the disk |
| LABEL | VARCHAR2(31) | Disk label portion of the name returned by discovery |
| PATH | VARCHAR2(256) | Operating system path name portion of the name returned by discovery |
| UDID | VARCHAR2(64) | Unique Device ID portion of the name returned by discovery |
| PRODUCT | VARCHAR2(32) | Name of the manufacturer and the name of the product. All disks with the same product id will have the same performance and reliability characteristics. |
| CREATE_DATE | DATE | Date and time when the disk was added to the disk group |
| MOUNT_DATE | DATE | Date and time when the disk was mounted by the first instance |
| REPAIR_TIMER | NUMBER | Seconds remaining until the disk is automatically dropped (`0` if not failed) |
| READS | NUMBER | Total number of I/O read requests for the disk |
| WRITES | NUMBER | Total number of I/O write requests for the disk |
| READ_ERRS | NUMBER | Total number of failed I/O read requests for the disk |
| WRITE_ERRS | NUMBER | Total number of failed I/O write requests for the disk |
| READ_TIMEOUT | NUMBER | Number of read I/Os that are timed out |
| WRITE_TIMEOUT | NUMBER | Number of write I/Os that are timed out |
| READ_TIME | NUMBER | Total I/O time (in seconds) for read requests for the disk if the `TIMED_STATISTICS` initialization parameter is set to `true` (`0` if set to `false`) |
| WRITE_TIME | NUMBER | Total I/O time (in seconds) for write requests for the disk if the `TIMED_STATISTICS` initialization parameter is set to `true` (`0` if set to `false`) |
| BYTES_READ | NUMBER | Total number of bytes read from the disk |
| BYTES_WRITTEN | NUMBER | Total number of bytes written to the disk |
| PREFERRED_READ | VARCHAR2(1) | Status of the preferred read failure group:<br>• `U` - Disk group has no preferred read failure group<br>• `Y` - Disk is a preferred read disk<br>• `N` - Disk is not a preferred read disk |

| Column | Datatype | Description |
|---|---|---|
| | | For a disk group with one or more preferred read failure groups, if the disk is in one of the preferred read failure groups, the value of this column is `Y`; otherwise it is `N`. |
| `HASH_VALUE` | `NUMBER` | A unique hash value for an Oracle ASM disk, computed using the Oracle ASM disk name (as specified by `NAME` column in `v$asm_disk`) and the Oracle ASM disk group name (as specified by `NAME` column in `v$asm_diskgroup`). |
| `HOT_READS` | `NUMBER` | Number of reads from the hot region on disk |
| `HOT_WRITES` | `NUMBER` | Number of writes to the hot region on disk |
| `HOT_BYTES_READ` | `NUMBER` | Number of bytes read from the hot region on disk |
| `HOT_BYTES_WRITTEN` | `NUMBER` | Number of bytes written to the hot region on disk |
| `COLD_READS` | `NUMBER` | Number of reads from the cold region on disk |
| `COLD_WRITES` | `NUMBER` | Number of writes to the cold region on disk |
| `COLD_BYTES_READ` | `NUMBER` | Number of bytes read from the cold region on disk |
| `COLD_BYTES_WRITTEN` | `NUMBER` | Number of bytes written to the cold region on disk |
| `VOTING_FILE` | `VARCHAR2(1)` | Indicates whether the disk contains a voting file (`Y`) or not (`N`) |
| `SECTOR_SIZE` | `NUMBER` | Physical block size (in bytes) |
| `FAILGROUP_TYPE` | `VARCHAR2(7)` | Type of the failure group:<br>• `REGULAR`<br>• `QUORUM` |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always `0`. |

**Note:**

The `GROUP_NUMBER` and `DISK_NUMBER` columns will only be valid if the disk is part of a disk group which is currently mounted by the instance. Otherwise, `GROUP_NUMBER` will be `0`, and `DISK_NUMBER` will be a unique value with respect to the other disks that also have a group number of `0`.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_DISK_IOSTAT

V$ASM_DISK_IOSTAT displays information about disk I/O statistics for each Oracle Automatic Storage Management (Oracle ASM) client. If this view is queried from the database instance, only the rows for that instance are shown.

| Column | Datatype | Description |
|---|---|---|
| INSTNAME | VARCHAR2(64) | Identifier for the DB/ADVM instance client |
| DBNAME | VARCHAR2(8) | Unique database name (DB_UNIQUE_NAME) |
| CLUSTERNAME | VARCHAR2(31) | Name of the cluster |
| GROUP_NUMBER | NUMBER | Number of the disk group containing the disk |
| DISK_NUMBER | NUMBER | Number assigned to the disk within its disk group |
| FAILGROUP | VARCHAR2(30) | Name of the failure group to which the disk belongs |
| READS | NUMBER | Total number of I/O read requests for the disk |
| WRITES | NUMBER | Total number of I/O write requests for the disk |
| READ_ERRS | NUMBER | Total number of failed I/O read requests for the disk |
| WRITE_ERRS | NUMBER | Total number of failed I/O write requests for the disk |
| READ_TIMEOUT | NUMBER | Number of read I/Os that are timed out |
| WRITE_TIMEOUT | NUMBER | Number of write I/Os that are timed out |
| READ_TIME | NUMBER | Total I/O time (in seconds) for read requests for the disk if the TIMED_STATISTICS initialization parameter is set to TRUE (0 if set to FALSE). |
| WRITE_TIME | NUMBER | Total I/O time (in seconds) for write requests for the disk if the TIMED_STATISTICS initialization parameter is set to TRUE (0 if set to FALSE) |
| BYTES_READ | NUMBER | Total number of bytes read from the disk |
| BYTES_WRITTEN | NUMBER | Total number of bytes written from the disk |
| HOT_READS | NUMBER | Number of reads from the hot region on disk |
| HOT_WRITES | NUMBER | Number of writes to the hot region on disk |
| HOT_BYTES_READ | NUMBER | Number of bytes read from the hot region on disk |
| HOT_BYTES_WRITTEN | NUMBER | Number of bytes written to the hot region on disk |
| COLD_READS | NUMBER | Number of reads from the cold region on disk |
| COLD_WRITES | NUMBER | Number of writes to the cold region on disk |
| COLD_BYTES_READ | NUMBER | Number of bytes read from the cold region on disk |
| COLD_BYTES_WRITTEN | NUMBER | Number of bytes written to the cold region on disk |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_DISK_STAT

V$ASM_DISK_STAT displays performance statistics in the same way that V$ASM_DISK does, but without performing discovery of new disks. This results in a less expensive operation. However, since discovery is not performed, the output of this view does not include any data about disks that are new to the system.

The columns for V$ASM_DISK_STAT are the same as those for V$ASM_DISK.

**See Also:**

"V$ASM_DISK"

# V$ASM_DISKGROUP

V$ASM_DISKGROUP displays one row for every Oracle Automatic Storage Management (Oracle ASM) disk group discovered by the Oracle ASM instance on the node.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Cluster-wide number assigned to the disk group (primary key) |
| NAME | VARCHAR2(30) | Name of the disk group |
| SECTOR_SIZE | NUMBER | Physical block size (in bytes) |
| BLOCK_SIZE | NUMBER | Oracle ASM metadata block size (in bytes) |
| ALLOCATION_UNIT_SIZE | NUMBER | Size of the allocation unit (in bytes) |
| STATE | VARCHAR2(11) | State of the disk group relative to the instance:<br>• CONNECTED - Disk group is in use by the database instance<br>• BROKEN - Database instance lost connectivity to the Oracle ASM instance that mounted the disk group<br>• UNKNOWN - Oracle ASM instance has never attempted to mount the disk group |

| Column | Datatype | Description |
|---|---|---|
| | | • DISMOUNTED - Disk group was cleanly dismounted by the Oracle ASM instance following a successful mount<br>• MOUNTED - Instance is successfully serving the disk group to its database clients<br>• QUIESCING - CRSCTL utility attempted to dismount a disk group that contains the Oracle Cluster Registry (OCR). The disk group cannot be dismounted until Cluster Ready Services (CRS) exits, because the disk group contains the OCR. |
| TYPE | VARCHAR2(6) | Redundancy type for the disk group:<br>• EXTERN<br>• NORMAL<br>• HIGH |
| TOTAL_MB | NUMBER | Total capacity of the disk group (in megabytes) |
| FREE_MB | NUMBER | Unused capacity of the disk group (in megabytes) |
| HOT_USED_MB | NUMBER | Number of used megabytes in the hot region |
| COLD_USED_MB | NUMBER | Number of used megabytes in the cold region |
| REQUIRED_MIRROR_FREE_MB | NUMBER | Amount of space that is required to be available in a given disk group in order to restore redundancy after one or more disk failures. The amount of space displayed in this column takes mirroring effects into account. |
| USABLE_FILE_MB | NUMBER | Amount of free space that can be safely utilized taking mirroring into account and yet be able to restore redundancy after a disk failure |
| OFFLINE_DISKS | NUMBER | Number of disks in the disk group that are currently offline |
| COMPATIBILITY | VARCHAR2(60) | Minimum software version required for an Oracle ASM instance to mount this disk group |
| DATABASE_COMPATIBILITY | VARCHAR2(60) | Minimum software version required for a database instance to use files in this disk group |
| VOTING_FILES | VARCHAR2(1) | Indicates whether the disk contains voting files (Y) or not (N) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**Note:**

The GROUP_NUMBER, TOTAL_MB, and FREE_MB columns are only meaningful if the disk group is mounted by the instance. Otherwise, their values will be 0.

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_DISKGROUP_STAT

V$ASM_DISKGROUP_STAT displays performance statistics in the same way that V$ASM_DISKGROUP does, but without performing discovery of new disk groups. This results in a less expensive operation. However, since discovery is not performed, the output of this view does not include any data about disk groups that are new to the system.

The columns for V$ASM_DISKGROUP_STAT are the same as those for V$ASM_DISKGROUP.

**See Also:**

"V$ASM_DISKGROUP"

# V$ASM_ESTIMATE

V$ASM_ESTIMATE displays an estimate of the work involved in execution plans for Oracle Automatic Storage Management (Oracle ASM) disk group rebalance and resync operations.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Oracle ASM disk group number |
| STATEMENT_ID | VARCHAR2(30) | Value of the optional STATEMENT_ID parameter specified in the EXPLAIN WORK statement |
| TIMESTAMP | DATE | Date and time when the EXPLAIN WORK statement was issued |
| EST_WORK | NUMBER | Estimated number of Allocation Units that have to be moved by the operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_FILE

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_FILE displays one row for each file in each disk group mounted by the Oracle ASM instance. For example, if there are three disk groups and five files in each, fifteen rows are displayed (unless the query is qualified with a WHERE clause).

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Number of the disk group containing the file (composite primary key) |
| FILE_NUMBER | NUMBER | Number of the file within the disk group (composite primary key) |
| COMPOUND_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and a file number in the low-order 24 bits (for efficient access to the view) |
| INCARNATION | NUMBER | Incarnation number for the file (composite primary key) |
| BLOCK_SIZE | NUMBER | Block size of the file (in bytes) |
| BLOCKS | NUMBER | Number of blocks in the file |
| BYTES | NUMBER | Number of bytes in the file |
| SPACE | NUMBER | Number of bytes allocated to the file |
| TYPE | VARCHAR2(64) | Type of the file. Possible values are as follows:<br>• CONTROLFILE<br>• DATAFILE<br>• ONLINELOG<br>• ARCHIVELOG<br>• TEMPFILE<br>• BACKUPSET<br>• XTRANSPORT<br>• PARAMETERFILE<br>• DATAGUARDCONFIG<br>• FLASHBACK<br>• CHANGETRACKING<br>• DUMPSET<br>• AUTOBACKUP |
| REDUNDANCY | VARCHAR2(6) | Redundancy of the file:<br>• UNPROT<br>• MIRROR<br>• HIGH |
| STRIPED | VARCHAR2(6) | Indicates how the file is striped:<br>• FINE<br>• COARSE |
| CREATION_DATE | DATE | Date on which the file was created |
| MODIFICATION_DATE | DATE | Date of the last open/close for writing, rounded back to the nearest hour |

| Column | Datatype | Description |
|---|---|---|
| REDUNDANCY_LOWERED | VARCHAR2(1) | Indicates whether a file has lower redundancy than what was expected (Y) or not (N). Redundancy is said to have been lowered for a file when one or more data extents in that file are not mirrored at the level specified by the administrator. In case of unprotected files, data extents could be missing altogether. Another possible value for this column is (U), which means that it is unknown.<br>This column is deprecated, and it always displays a value of U. |
| PERMISSIONS | VARCHAR2(16) | Access permissions of the file, in order of user, group, and other |
| USER_NUMBER | NUMBER | User number |
| USER_INCARNATION | NUMBER | Incarnation number of the user |
| USERGROUP_NUMBER | NUMBER | User group number |
| USERGROUP_INCARNATION | NUMBER | Incarnation number of the user group |
| PRIMARY_REGION | VARCHAR2(4) | Region used for allocating primary extents:<br>• HOT<br>• COLD |
| MIRROR_REGION | VARCHAR2(4) | Region for allocating mirrored extents:<br>• HOT<br>• COLD |
| HOT_READS | NUMBER | Number of reads from the hot region for the file |
| HOT_WRITES | NUMBER | Number of writes to the hot region for the file |
| HOT_BYTES_READ | NUMBER | Number of bytes read from the hot region for the file |
| HOT_BYTES_WRITTEN | NUMBER | Number of bytes written to the hot region for the file |
| COLD_READS | NUMBER | Number of reads from the cold region for the file |
| COLD_WRITES | NUMBER | Number of writes to the cold region for the file |
| COLD_BYTES_READ | NUMBER | Number of bytes read from the cold region for the file |
| COLD_BYTES_WRITTEN | NUMBER | Number of bytes written to the cold region for the file |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_FILESYSTEM

V$ASM_FILESYSTEM displays information for every mounted Oracle ACFS.

| Column | Datatype | Description |
|---|---|---|
| FS_NAME | VARCHAR2(1024) | Mount point (primary key) |
| AVAILABLE_TIME | DATE | Mount time or the time that the file system became available again; NULL if the file system is not available |
| BLOCK_SIZE | NUMBER | File system block size (in kilobytes) |
| STATE | VARCHAR2(13) | File system status::<br>• NOT AVAILABLE<br>• AVAILABLE<br>• OFFLINE - Either the Oracle ASM instance is down, the disk group has been forced dismounted, or less commonly, a metadata I/O failure occurred or serious metadata corruption was detected. In the case of a metadata I/O failure, the file system is also marked as corrupt. An offline file system can only be unmounted; other attempts at access result in errors. |
| CORRUPT | VARCHAR2(5) | Indicates whether the file system needs the fixer (fsck, acfschkdsk) to be run (TRUE) or not (FALSE) |
| NUM_VOL | NUMBER | Number of volumes in the file system |
| TOTAL_SIZE | NUMBER | Total capacity of the file system (in megabytes) |
| TOTAL_FREE | NUMBER | Total free space in the file system (in megabytes) |
| TOTAL_SNAP_SPACE_USAGE | NUMBER | Total space used by snapshots (in megabytes) |
| REPLSTATE | VARCHAR2(7) | Replication status:<br>• NO REPL - Replication is not initialized<br>• PRIMARY - File system is initialized for replication as a primary<br>• STANDBY - File system is initialized for replication as a standby |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_OPERATION

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_OPERATION displays one row for every active Oracle ASM long running operation executing in the Oracle ASM instance.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Disk group number (primary key). This is the foreign key to the V$ASM_DISKGROUP view. |
| OPERATION | CHAR(5) | Type of the operation:<br>• REBAL - Rebalance pending for this group. The disk group is rebalancing.<br>Starting with Oracle Database 12*c*, new queries should use the PASS column instead of this column. |
| PASS | VARCHAR2(9) | Type of the operation:<br>• REBALANCE - Rebalance pending for this group. The disk group is rebalancing.<br>• RESYNC - Resync operation in progress to bring one or more Oracle ASM disks online<br>• COMPACT - Oracle ASM is moving user data closer together, which improves performance by reducing seek distance<br>Starting with Oracle Database 12*c*, new queries should use this column instead of the OPERATION column. |
| STATE | VARCHAR2(4) | State of the operation:<br>• WAIT - No operations running for the group<br>• RUN - Operation running for the group<br>• REAP - Operation is being run down<br>• DONE - Displayed for a pass that is complete<br>• ERRS - Operation halted by errors |
| POWER | NUMBER | Power requested for the operation as specified by the ASM_POWER_LIMIT initialization parameter or command syntax |
| ACTUAL | NUMBER | Power allocated to the operation |
| SOFAR | NUMBER | Number of Allocation Units that have been moved so far by the operation |
| EST_WORK | NUMBER | Estimated number of Allocation Units that have to be moved by the operation |
| EST_RATE | NUMBER | Estimated number of Allocation Units that are being moved per minute by the operation |
| EST_MINUTES | NUMBER | Estimated amount of time (in minutes) that the remainder of the operation is expected to take |
| ERROR_CODE | VARCHAR2(44) | Oracle external error code; NULL if no error |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_TEMPLATE

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_TEMPLATE displays one row for every template present in every disk group mounted by the Oracle ASM instance. In a database instance, V$ASM_TEMPLATE displays one row for every template present in every disk group mounted by the Oracle ASM instance with which the database instance communicates.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Owning disk group number (foreign key to the V$ASM_DISKGROUP view) |
| ENTRY_NUMBER | NUMBER | Template number (primary key) |
| REDUNDANCY | VARCHAR2(6) | Redundancy of the template:<br>• UNPROT<br>• MIRROR<br>• HIGH |
| STRIPE | VARCHAR2(6) | Indicates how the template is striped:<br>• FINE<br>• COARSE |
| SYSTEM | VARCHAR2(1) | Indicates whether the template is a system template (Y) or not (N) |
| NAME | VARCHAR2(30) | Name of the template |
| PRIMARY_REGION | VARCHAR2(4) | Region used for allocating primary extents:<br>• HOT<br>• COLD |
| MIRROR_REGION | VARCHAR2(4) | Region for allocating mirrored extents:<br>• HOT<br>• COLD |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_USER

V$ASM_USER displays the effective operating system user names of connected database instances and of file owners.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Oracle ASM disk group number |
| USER_NUMBER | NUMBER | Oracle ASM internal unique user number |
| COMPOUND_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and a user number in the low-order 24 bits (for efficient access to the view) |
| INCARNATION | NUMBER | Incarnation number of the user |
| CLUSTER_ID | VARCHAR2(128) | Oracle ASM cluster ID. The CLUSTER_ID and OS_NAME pair can be used to uniquely identify a user. |
| OS_ID | VARCHAR2(128) | Operating system user ID |
| OS_NAME | VARCHAR2(33) | Operating system user name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_USERGROUP

V$ASM_USERGROUP displays the creator for each Oracle Automatic Storage Management (Oracle ASM) File Access Control group.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Oracle ASM disk group number |
| USERGROUP_NUMBER | NUMBER | Number of the user group |
| COMPOUND_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and a user group number in the low-order 24 bits (for efficient access to the view) |
| INCARNATION | NUMBER | Incarnation number of the user group |
| OWNER_NUMBER | NUMBER | User group owner identified by a unique number |
| OWNER_INCARNATION | NUMBER | Incarnation number of the user group owner |
| NAME | VARCHAR2(64) | User group name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_USERGROUP_MEMBER

V$ASM_USERGROUP_MEMBER displays the members for each Oracle Automatic Storage Management (Oracle ASM) File Access Control group.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Oracle ASM disk group number |
| MEMBER_NUMBER | NUMBER | Number of the user group member |
| MEMBER_INCARNATION | NUMBER | Incarnation number of the user group member |
| USERGROUP_NUMBER | NUMBER | User group number |
| USERGROUP_INCARNATION | NUMBER | Incarnation number of the user group |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_VOLUME

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_VOLUME displays information about each Oracle ADVM volume.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Cluster-wide number assigned to the disk group (composite primary key) |
| VOLUME_NAME | VARCHAR2(30) | Name of the volume |
| COMPOUND_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and a volume number in the low-order 24 bits (for efficient access to the view) |
| SIZE_MB | NUMBER | Size of the volume (in megabytes) |
| VOLUME_NUMBER | NUMBER | Number of the Volume within the disk group (composite primary key) |
| REDUNDANCY | VARCHAR2(6) | Redundancy type for the volume:<br>• UNPROT<br>• HIGH<br>• MIRROR |
| STRIPE_COLUMNS | NUMBER | Number of columns in a stripe set |
| STRIPE_WIDTH_K | NUMBER | Stripe width of the volume (in kilobytes) |
| STATE | VARCHAR2(8) | Indicates whether the volume is enabled (ENABLED), disabled (DISABLED), or remote (REMOTE) |
| FILE_NUMBER | NUMBER | Volume file number |
| INCARNATION | NUMBER | Volume file incarnation number |
| DRL_FILE_NUMBER | NUMBER | Volume Dirty Region Logging (DRL) file used for mirrored volumes |

| Column | Datatype | Description |
|---|---|---|
| RESIZE_UNIT_MB | NUMBER | Volume allocation unit (in megabytes) that a volume can be created |
| USAGE | VARCHAR2(30) | Optional usage string for the volume |
| VOLUME_DEVICE | VARCHAR2(256) | OSD path for the volume device |
| MOUNTPATH | VARCHAR2(1024) | Optional mount path string for the volume |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$ASM_VOLUME_STAT

In an Oracle Automatic Storage Management (Oracle ASM) instance, V$ASM_VOLUME_STAT displays information about statistics for each Oracle ADVM volume.

| Column | Datatype | Description |
|---|---|---|
| GROUP_NUMBER | NUMBER | Cluster-wide number assigned to the disk group (composite primary key) |
| VOLUME_NAME | VARCHAR2(30) | Name of the volume |
| COMPOUND_INDEX | NUMBER | A 32-bit number consisting of a disk group number in the high-order 8 bits and a volume number in the low-order 24 bits (for efficient access to the view) |
| VOLUME_NUMBER | NUMBER | Number of the Volume within the disk group (composite primary key) |
| READS | NUMBER | Total number of read requests for this volume |
| WRITES | NUMBER | Total number of write requests for this volume |
| READ_ERRS | NUMBER | Total number of failed read I/O operations for this volume |
| WRITE_ERRS | NUMBER | Total number of failed write I/O operations for this volume |
| READ_TIME | NUMBER | Total I/O time (in seconds) for read requests for this volume |
| WRITE_TIME | NUMBER | Total I/O time (in seconds) for write requests for this volume |

| Column | Datatype | Description |
|---|---|---|
| BYTES_READ | NUMBER | Total number of bytes read for this volume |
| BYTES_WRITTEN | NUMBER | Total number of bytes written for this volume |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>For this view, the value is always 0. |

**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* for additional information about using views to display Oracle ASM information

# V$AW_AGGREGATE_OP

V$AW_AGGREGATE_OP displays the aggregation operators available in analytic workspaces. You can use this view in an application to provide a list of choices.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(14) | Keyword for the aggregation operator |
| LONGNAME | VARCHAR2(30) | Descriptive name for the operator |
| DEFAULT_WEIGHT | NUMBER | Default weight factor for weighted operators |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$AW_ALLOCATE_OP

V$AW_ALLOCATE_OP displays the allocation operators available in analytic workspaces. You can use this view in an application to provide a list of choices.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(14) | Keyword for the allocation operator |
| LONGNAME | VARCHAR2(30) | Descriptive name for the operator |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$AW_CALC

`V$AW_CALC` reports on the effectiveness of various caches used by Oracle OLAP during dynamic aggregation.

| Column | Datatype | Description |
|---|---|---|
| `SESSION_ID` | `NUMBER` | A unique numeric identifier for the session. |
| `AGGREGATE_CACHE_HITS` | `NUMBER` | The number of times a dimension member is found in the aggregate cache (a hit).<br>The number of hits for run-time aggregation can be increased by fetching data across the dense dimension |
| `AGGREGATE_CACHE_MISSES` | `NUMBER` | The number of times a dimension member is not found in the aggregate cache and must be read from disk (a miss). |
| `SESSION_CACHE_HITS` | `NUMBER` | The number of times the data is found in the session cache (a hit). |
| `SESSION_CACHE_MISSES` | `NUMBER` | The number of times the data is not found in the session cache (a miss). |
| `POOL_HITS` | `NUMBER` | The number of times the data is found in a page in the OLAP page pool (a hit). |
| `POOL_MISSES` | `NUMBER` | The number of times the data is not found in the OLAP page pool (a miss). |
| `POOL_NEW_PAGES` | `NUMBER` | The number of newly created pages in the OLAP page pool that have not yet been written to the workspace LOB. |
| `POOL_RECLAIMED_PAGES` | `NUMBER` | The number of previously unused pages that have been recycled with new data. |
| `CACHE_WRITES` | `NUMBER` | The number of times the data from the OLAP page pool has been written to the database cache. |
| `POOL_SIZE` | `NUMBER` | The number of kilobytes in the OLAP page pool. |
| `CURR_DML_COMMAND` | `VARCHAR2(64)` | The command currently being executed. |
| `PREV_DML_COMMAND` | `VARCHAR2(64)` | The command most recently completed. |
| `AGGR_FUNC_LOGICAL_NA` | `NUMBER` | The number of times the aggregation engine returns a logical NA because the AGGINDEX option is on and the composite tuple does not exist. |
| `AGGR_FUNC_PRECOMPUTE` | `NUMBER` | The number of times the aggregation engine finds a value in a position that it was called to calculate. |

| Column | Datatype | Description |
|---|---|---|
| AGGR_FUNC_CALCS | NUMBER | The number of times the aggregation engine calculates a parent value based on the values of its children. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$AW_LONGOPS

V$AW_LONGOPS displays status information about active SQL cursors initiated in an analytic workspace.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Identifier for the session in which the fetch is executing. This table can be joined with V$SESSION to obtain the user name. |
| CURSOR_NAME | VARCHAR2(64) | Name assigned to the cursor |
| COMMAND | VARCHAR2(17) | Command that is actively fetching data from relational tables:<br>• QUERY<br>• FETCH<br>• IMPORT<br>• EXECUTE<br>• UPDATE<br>• SOLVE<br>• CLEAR<br>• LOAD CUBE<br>• LOAD DIMENSION<br>• DIMENSION COMPILE |
| STATUS | VARCHAR2(9) | Status of the current operation:<br>• EXECUTING - Command has begun executing<br>• FETCHING - Data is being fetched into the analytic workspace<br>• FINISHED - Command has finished executing. This status appears very briefly before the record disappears from the table. |
| ROWS_PROCESSED | NUMBER | Number of rows already inserted, updated, or deleted |
| SEQ_NUMBER | NUMBER | Sequence number in the Cube Build log |
| SQL_ID | VARCHAR2(13) | SQL ID of the statement |
| TARGET | VARCHAR2(64) | Operated object name |
| TARGET_DESC | VARCHAR2(64) | A brief description of the operated object |
| START_TIME | DATE | Time the command started executing |
| LAST_UPDATE_TIME | DATE | Last updated time |

| Column | Datatype | Description |
|---|---|---|
| ELAPSED_SECONDS | NUMBER | Number of seconds between START_TIME and LAST_UPDATE_TIME |
| SOFAR | NUMBER | Number of units so far |
| TOTALWORK | NUMBER | Total number of units |
| UNITS | VARCHAR2(6) | Units description:<br>• ROWS<br>• NODES<br>• VALUES |
| MESSAGE | VARCHAR2(512) | Message for the user |
| USERNAME | VARCHAR2(32) | User name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$AW_OLAP

V$AW_OLAP provides a record of active sessions and their use with analytic workspaces. A row is generated whenever an analytic workspace is created or attached. The first row for a session is created when the first command is issued. It identifies the SYS.EXPRESS workspace, which is attached automatically to each session. Rows related to a particular analytic workspace are deleted when the workspace is detached from the session or the session ends.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | A unique numeric identifier for a session |
| AW_NUMBER | NUMBER | A unique numeric identifier for an analytic workspace. To get the name of the analytic workspace, join this column to the AW_NUMBER column of the USER_AWS view or to the AWSEQ# column of the AW$ table. |
| ATTACH_MODE | VARCHAR2(10) | READ ONLY or READ WRITE |
| GENERATION | NUMBER | The generation of an analytic workspace. Each UPDATE creates a new generation. Sessions attaching the same workspace between UPDATE commands share the same generation. |
| TEMP_SPACE_PAGES | NUMBER | The number of pages stored in temporary segments for the analytic workspace. |
| TEMP_SPACE_READS | NUMBER | The number of times data has been read from a temporary segment and not from the page pool. |
| LOB_READS | NUMBER | The number of times data has been read from the table where the analytic workspace is stored (the permanent LOB). |

| Column | Datatype | Description |
|---|---|---|
| POOL_CHANGED_PAGES | NUMBER | The number of pages in the page pool that have been modified in this analytic workspace. |
| POOL_UNCHANGED_PAGES | NUMBER | The number of pages in the page pool that have not been modified in this analytic workspace. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$AW_SESSION_INFO

V$AW_SESSION_INFO provides information about each active session. A transaction is a single exchange between a client session and Oracle OLAP. Multiple commands can execute within a single transaction.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | A unique numeric identifier for a session |
| CLIENT_TYPE | VARCHAR2(64) | OLAP |
| SESSION_STATE | VARCHAR2(64) | TRANSACTING, NOT_TRANSACTING, EXCEPTION_HANDLING, CONSTRUCTING, CONSTRUCTED, DECONSTRUCTING, or DECONSTRUCTED |
| SESSION_HANDLE | NUMBER | The session identifier |
| USERID | VARCHAR2(64) | The database user name under which the session opened |
| TOTAL_TRANSACTION | NUMBER | The total number of transactions executed within the session; this number provides a general indication of the level of activity in the session |
| TRANSACTION_TIME | NUMBER | The elapsed time in milliseconds of the mostly recently completed transaction |
| TOTAL_TRANSACTION_TIME | NUMBER | The total elapsed time in milliseconds in which transactions were being executed |
| AVERAGE_TRANSACTION_TIME | NUMBER | The average elapsed time in milliseconds to complete a transaction |
| TRANSACTION_CPU_TIME | NUMBER | The total CPU time in milliseconds used to complete the most recent transaction |
| TOTAL_TRANSACTION_CPU_TIME | NUMBER | The total CPU time used to execute all transactions in this session; this total does not include transactions that are currently in progress |
| AVERAGE_TRANSACTION_CPU_TIME | NUMBER | The average CPU time to complete a transaction; this average does not include transactions that are currently in progress |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP

V$BACKUP displays the backup status of all online data files.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | File identifier |
| STATUS | VARCHAR2(18) | File status: NOT ACTIVE, ACTIVE (backup in progress), OFFLINE NORMAL, or description of an error.<br>NOT ACTIVE indicates that the file is not currently in backup mode (that is, an ALTER TABLESPACE ... BEGIN BACKUP or ALTER DATABASE BEGIN BACKUP statement has not been issued), whereas ACTIVE indicates that the file is currently in backup mode. |
| CHANGE# | NUMBER | System change number when backup started |
| TIME | DATE | Time the backup started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_ARCHIVELOG_DETAILS

V$BACKUP_ARCHIVELOG_DETAILS contains information about all restorable archive logs. It will include all archived logs backed up in a backup set or proxy copies.

| Column | Datatype | Description |
|---|---|---|
| BTYPE | CHAR(9) | Backup type container, either BACKUPSET or PROXYCOPY |
| BTYPE_KEY | NUMBER | Unique identifier for the backup type. For BACKUPSET, it is BS_KEY. |
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session recid |
| SESSION_STAMP | NUMBER | Session stamp |

| Column | Datatype | Description |
|---|---|---|
| ID1 | NUMBER | If BACKUPSET, it contains SET_STAMP<br>If PROXYCOPY, it is RECID from the control file |
| ID2 | NUMBER | If BACKUPSET, it contains SET_COUNT<br>If PROXYCOPY, it is STAMP |
| THREAD# | NUMBER | Thread number |
| SEQUENCE# | NUMBER | Sequence number |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change SCN |
| RESETLOGS_TIME | DATE | Resetlogs change time |
| FIRST_CHANGE# | NUMBER | First change SCN |
| FIRST_TIME | DATE | First change time |
| NEXT_CHANGE# | NUMBER | Next change SCN |
| NEXT_TIME | DATE | Next change time |
| FILESIZE | NUMBER | File size |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the archive log and the blocks that RMAN backed up. This is *not* the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| FILESIZE_DISPLAY | VARCHAR2(4000) | Same value as the FILESIZE column, but converted to a user-displayable format, for example *n*M, *n*G, *n*T, *n*P, and so on |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_ARCHIVELOG_SUMMARY

V$BACKUP_ARCHIVELOG_SUMMARY provides archive log summary information based on archive logs in the backup set or on proxy copies.

| Column | Datatype | Description |
|---|---|---|
| NUM_FILES_BACKED | NUMBER | Number of files backed up |
| NUM_DISTINCT_FILES_BACKED | NUMBER | Number of distinct archive log files backed up |
| MIN_FIRST_CHANGE# | NUMBER | Lowest SCN range value |
| MAX_NEXT_CHANGE# | NUMBER | Highest SCN range value |
| MIN_FIRST_TIME | DATE | Lowest SCN range time |

| Column | Datatype | Description |
|---|---|---|
| MAX_NEXT_TIME | DATE | Highest SCN range time |
| INPUT_BYTES | NUMBER | Total input bytes read |
| OUTPUT_BYTES | NUMBER | Output size of backups |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the archive log and the blocks that RMAN backed up. This is *not* the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for input bytes |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_ASYNC_IO

V$BACKUP_ASYNC_IO displays performance information about ongoing and recently completed RMAN backups and restores. For each backup, it contains one row for each input data file, one row for the aggregate total performance of all data files, and one row for the output backup piece. This data is not stored persistently, and is not preserved when the instance is re-started.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Oracle SID of the session doing the backup or restore |
| SERIAL | NUMBER | Use count for the SID doing the backup or restore |
| USE_COUNT | NUMBER | A counter that can be used to identify rows from different backup sets |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS record stamp |
| DEVICE_TYPE | VARCHAR2(17) | Device type where the file is located |
| TYPE | VARCHAR2(9) | INPUT, OUTPUT, or AGGREGATE |
| STATUS | VARCHAR2(11) | NOT STARTED, IN PROGRESS, or FINISHED |
| FILENAME | VARCHAR2(513) | Name of the backup file being read or written |
| SET_COUNT | NUMBER | Set count of the backup set being read or written |
| SET_STAMP | NUMBER | Set stamp of the backup set being read or written |

| Column | Datatype | Description |
|---|---|---|
| BUFFER_SIZE | NUMBER | Size of the buffers being used to read/write the file, in bytes |
| BUFFER_COUNT | NUMBER | Number of buffers being used to read/write the file |
| TOTAL_BYTES | NUMBER | Total number of bytes that will be read or written for the file, if known. If not known, this column will be null |
| OPEN_TIME | DATE | Time the file was opened. If TYPE='AGGREGATE', then this is the time that the first file in the aggregate was opened |
| CLOSE_TIME | DATE | Time the file was closed. If TYPE='AGGREGATE', then this is the time that the last file in the aggregate was closed |
| ELAPSED_TIME | NUMBER | Time, in hundredths of a second, that the file was open |
| MAXOPENFILES | NUMBER | Number of concurrently open DISK files. This value is only present in rows where TYPE='AGGREGATE'. |
| BYTES | NUMBER | Number of bytes read or written so far |
| EFFECTIVE_BYTES_PER_SECOND | NUMBER | I/O rate that was achieved with this device during this backup |
| IO_COUNT | NUMBER | Number of I/Os that were performed to this file |
| READY | NUMBER | Number of asynchronous requests for which a buffer was immediately ready for use |
| SHORT_WAITS | NUMBER | Number of times that a buffer was not immediately available, but a buffer became available after doing a nonblocking poll for I/O completion |
| SHORT_WAIT_TIME_TOTAL | NUMBER | Total time, in hundredths of a second, taken by nonblocking polls for I/O completion |
| SHORT_WAIT_TIME_MAX | NUMBER | Maximum time taken for a nonblocking poll for I/O completion, in hundredths of a second |
| LONG_WAITS | NUMBER | The number of times that a buffer was not immediately available, and only became available after a blocking wait was issued |
| LONG_WAIT_TIME_TOTAL | NUMBER | The total time, in hundredths of a second, taken by blocking waits for I/O completion |
| LONG_WAIT_TIME_MAX | NUMBER | The maximum time taken for a blocking wait for I/O completion, in hundredths of a second |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_CONTROLFILE_DETAILS

V$BACKUP_CONTROLFILE_DETAILS contains information about restorable control files. It will include all the control files backed up in the backup set, image copies, and proxy copies.

| Column | Datatype | Description |
|---|---|---|
| BTYPE | CHAR(9) | Backup type container. Possible values are BACKUPSET, IMAGECOPY, PROXYCOPY. |
| BTYPE_KEY | NUMBER | Unique identifier for the backup type, either BS_KEY or COPY_KEY. |
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session recid |
| SESSION_STAMP | NUMBER | Session stamp |
| ID1 | NUMBER | If BACKUPSET, it contains SET_STAMP<br>If IMAGECOPY or PROXYCOPY, it is RECID from the control file |
| ID2 | NUMBER | If BACKUPSET, it contains SET_COUNT<br>If IMAGECOPY or PROXYCOPY, it is STAMP |
| CREATION_TIME | DATE | File creation time |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change SCN |
| RESETLOGS_TIME | DATE | Resetlogs change time |
| CHECKPOINT_CHANGE# | NUMBER | Checkpoint change SCN |
| CHECKPOINT_TIME | DATE | Checkpoint change time |
| FILESIZE | NUMBER | File size, in bytes, for the output of backing up this control file |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the datafile and the blocks that RMAN backed up. This is *not* the the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| FILESIZE_DISPLAY | VARCHAR2(4000) | Same value as the FILESIZE column, but converted to a user-displayable format, for example *n*M, *n*G, *n*T, *n*P, and so on |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_CONTROLFILE_SUMMARY

V$BACKUP_CONTROLFILE_SUMMARY provides control file summary information, based on either a backup set of files, image copies, or proxy copies.

| Column | Datatype | Description |
|---|---|---|
| NUM_FILES_BACKED | NUMBER | Number of files backed up for specific criteria |
| NUM_DISTINCT_FILES_BACKED | NUMBER | Number of distinct files backed up |
| MIN_CHECKPOINT_CHANGE# | NUMBER | Minimum checkpoint change number of the data file for specified criteria |
| MAX_CHECKPOINT_CHANGE# | NUMBER | Maximum checkpoint change number of the data file for specified criteria |
| MIN_CHECKPOINT_TIME | DATE | Minimum checkpoint time of the data file for specified criteria |
| MAX_CHECKPOINT_TIME | DATE | Maximum checkpoint time of the data file for specified criteria |
| INPUT_BYTES | NUMBER | Total input bytes of files read |
| OUTPUT_BYTES | NUMBER | Total output bytes written |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the datafile and the blocks that RMAN backed up. This is *not* the the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for input bytes |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_COPY_DETAILS

V$BACKUP_COPY_DETAILS contains information about all available control file and data file copies.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session recid |
| SESSION_STAMP | NUMBER | Session stamp |
| COPY_KEY | NUMBER | Unique identifier for this data file or control file copy |
| FILE# | NUMBER | Absolute data file number |

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(513) | File name of the data file copy. The maximum length of the name is dependent on your operating system. |
| TAG | VARCHAR2(32) | Data file copy tag |
| CREATION_CHANGE# | NUMBER | Data file creation change number |
| CREATION_TIME | DATE | Data file creation timestamp |
| CHECKPOINT_CHANGE# | NUMBER | Checkpoint change number of the data file when the copy was made |
| CHECKPOINT_TIME | DATE | Checkpoint timestamp of the data file when the copy was made |
| MARKED_CORRUPT | NUMBER | Number of blocks marked corrupt by this copy operation. That is, blocks that were not marked corrupted in the source data file, but were detected and marked as corrupted during the copy operation. |
| OUTPUT_BYTES | NUMBER | Total output bytes written |
| COMPLETION_TIME | DATE | Time when the copy was completed |
| CONTROLFILE_TYPE | VARCHAR2(1) | Type of control file. B indicates normal copies. S indicates standby copies. |
| KEEP | VARCHAR2(3) | (YES \| NO) Indicates whether or not this backup set has a retention policy that is different than the value for the configure retention policy |
| KEEP_UNTIL | DATE | If specified, this is the date after which the backup becomes obsolete. If this column is null, then the backup never expires. |
| KEEP_OPTIONS | VARCHAR2(11) | Lists additional retention options for this backup set. Possible values are:<br>• LOGS - The logs needed to recover this backup set are kept<br>• NOLOGS - The logs needed to recover this backup set are not kept |
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether the file was created in the fast recovery area (YES) or not (NO) |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Size of backup set to display |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_COPY_SUMMARY

V$BACKUP_COPY_SUMMARY provides summary information for the output data file and control file copy.

| Column | Datatype | Description |
|---|---|---|
| NUM_COPIES | NUMBER | Number of copies created |
| NUM_DISTINCT_COPIES | NUMBER | Number of distinct copies (that contain data files with different checkpoints) |
| MIN_CHECKPOINT_CHANGE# | NUMBER | Minimum checkpoint change SCN |
| MAX_CHECKPOINT_CHANGE# | NUMBER | Maximum checkpoint change SCN |
| MIN_CHECKPOINT_TIME | DATE | Minimum checkpoint change time |
| MAX_CHECKPOINT_TIME | DATE | Maximum checkpoint change time |
| OUTPUT_BYTES | NUMBER | Total number of output bytes |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_CORRUPTION

V$BACKUP_CORRUPTION displays information about corrupt block ranges in data file backups from the control file. Note that corruptions are not tolerated in the control file and archived redo log backups.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Backup corruption record ID |
| STAMP | NUMBER | Backup corruption record stamp |
| SET_STAMP | NUMBER | Backup set stamp |
| SET_COUNT | NUMBER | Backup set count |
| PIECE# | NUMBER | backup piece that contains this corrupt block |
| FILE# | NUMBER | Absolute file number of the data file that contains the corrupt blocks |
| BLOCK# | NUMBER | Block number of the first corrupt block in the range of corrupted blocks |
| BLOCKS | NUMBER | Number of corrupted blocks found starting with BLOCK# |

| Column | Datatype | Description |
|---|---|---|
| CORRUPTION_CHANGE# | NUMBER | Change number at which the logical corruption was detected. Set to 0 to indicate media corruption. |
| MARKED_CORRUPT | VARCHAR2(3) | Indicates whether this corruption was not previously detected by the Oracle Database (YES) or the Oracle Database had already discovered this corrupt block and marked it as corrupt (NO). Note that when a corrupt block is encountered in a backup, and was not already marked corrupt by the Oracle Database, then the backup process does not mark the block as corrupt in the production data file. Thus, this field may be YES for the same block in more than one backup set. |
| CORRUPTION_TYPE | VARCHAR2(9) | Type of block corruption in the data file:<br>• ALL ZERO - Block header on disk contained only zeros. The block may be valid if it was never filled and if it is in an Oracle7 file. The buffer will be reformatted to the Oracle8 standard for an empty block.<br>• FRACTURED - Block header looks reasonable, but the front and back of the block are different versions.<br>• CHECKSUM - optional check value shows that the block is not self-consistent. It is impossible to determine exactly why the check value fails, but it probably fails because sectors in the middle of the block are from different versions.<br>• CORRUPT - Block is wrongly identified or is not a data block (for example, the data block address is missing)<br>• LOGICAL - Block is logically corrupt<br>• NOLOGGING - Block does not have redo log entries (for example, NOLOGGING operations on primary database can introduce this type of corruption on a physical standby) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_DATAFILE

V$BACKUP_DATAFILE displays information about control files and data files in backup sets from the control file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Backup data file record ID |
| STAMP | NUMBER | Backup data file record stamp |
| SET_STAMP | NUMBER | Backup set stamp |
| SET_COUNT | NUMBER | Backup set count |
| FILE# | NUMBER | Data file number; set to 0 for control file |
| CREATION_CHANGE# | NUMBER | Creation system change number (SCN) of the data file |

| Column | Datatype | Description |
|---|---|---|
| CREATION_TIME | DATE | Creation timestamp of the data file |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs system change number (SCN) of the data file when it was backed up |
| RESETLOGS_TIME | DATE | Resetlogs timestamp of the data file when it was backed up |
| INCREMENTAL_LEVEL | NUMBER | Normal full backups have a NULL value, level 0 incremental backups have a value of 0, and level 1 incremental backups have a value of 1 |
| INCREMENTAL_CHANGE# | NUMBER | All blocks changed after the incremental change number is included in this backup; set to 0 for a full backup |
| CHECKPOINT_CHANGE# | NUMBER | All changes up to the checkpoint change number are included in this backup |
| CHECKPOINT_TIME | DATE | Timestamp of the checkpoint |
| ABSOLUTE_FUZZY_CHANGE# | NUMBER | Highest change number in this backup |
| MARKED_CORRUPT | NUMBER | Number of blocks marked corrupt |
| MEDIA_CORRUPT | NUMBER | Number of blocks media corrupt |
| LOGICALLY_CORRUPT | NUMBER | Number of blocks logically corrupt |
| DATAFILE_BLOCKS | NUMBER | Size of the data file in blocks at backup time. This value is also the number of blocks taken by the data file restarted from this backup. |
| BLOCKS | NUMBER | Size of the backup data file (in blocks). Unused blocks are not copied to the backup. |
| BLOCK_SIZE | NUMBER | Block size |
| OLDEST_OFFLINE_RANGE | NUMBER | RECID of the oldest offline range record in this backup control file. 0 for data file backups. |
| COMPLETION_TIME | DATE | Time completed |
| CONTROLFILE_TYPE | VARCHAR2(1) | B - Normal copies<br>S - Standby copies |
| USED_CHANGE_TRACKING | VARCHAR2(3) | Indicates whether change tracking data was used to accelerate this incremental backup (YES) or whether change tracking data was not used (NO) |
| BLOCKS_READ | NUMBER | Number of blocks that were scanned while taking this backup. If this was an incremental backup, and change tracking was used to optimize the backup, then the value of this column will be smaller than DATAFILE_BLOCKS. Otherwise, the value of this column will be the same as DATAFILE_BLOCKS. Even when change tracking data is used, the value of this column may be larger than BLOCKS, because the data read by change tracking is further refined during the process of creating an incremental backup. |

| Column | Datatype | Description |
|---|---|---|
| USED_OPTIMIZATION | VARCHAR2(3) | Indicates whether backup optimization was applied (YES) or not (NO) |
| FOREIGN_DBID | NUMBER | Foreign DBID of the database from which this data file was transported. The value is 0 if the file backed up is not a foreign database file. |
| PLUGGED_READONLY | VARCHAR2(3) | YES if this is a backup of a transported read-only foreign file; otherwise NO. |
| PLUGIN_CHANGE# | NUMBER | SCN at which the foreign data file was transported into the database. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_CHANGE# | NUMBER | The SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_TIME | DATE | The time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| SECTION_SIZE | NUMBER | Specifies the number of blocks in each section of a multisection backup. Value is 0 for whole file backups. |
| UNDO_OPTIMIZED | VARCHAR2(3) | Indicates whether undo blocks were ignored when creating the backup data file (YES) or not (NO) |
| BLOCKS_SKIPPED_IN_CELL | NUMBER | Number of blocks that were not backed up because they were skipped by the Exadata cell<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| BACKED_BY_PDB | VARCHAR2(3) | Recovery Manager (RMAN) allows a PDB to be backed up in two ways. The value in this column indicates how the PDB backup was taken:<br>• YES: The backup was taken when connected to the PDB<br>• NO: The backup was taken when connected to the root container |

# V$BACKUP_DATAFILE_DETAILS

V$BACKUP_DATAFILE_DETAILS contains information about restorable data files. It will include all data files backed in the backup set, image copies, and proxy copies.

| Column | Datatype | Description |
|---|---|---|
| BTYPE | CHAR(9) | Backup type container. Possible values are: BACKUPSET, IMAGECOPY, PROXYCOPY. |

| Column | Datatype | Description |
|---|---|---|
| BTYPE_KEY | NUMBER | Unique identifier for the backup type. For BACKUPSET, it is BS_KEY. |
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session record ID |
| SESSION_STAMP | NUMBER | Session stamp |
| ID1 | NUMBER | If BACKUPSET, it contains SET_STAMP.<br>If IMAGECOPY or PROXYCOPY, it is RECID from the control file. |
| ID2 | NUMBER | If BACKUPSET, it contains SET_COUNT.<br>If IMAGECOPY or PROXYCOPY, it is STAMP. |
| FILE# | NUMBER | File number |
| CREATION_CHANGE# | NUMBER | File creation change SCN |
| CREATION_TIME | DATE | File creation time |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change SCN |
| RESETLOGS_TIME | DATE | Resetlogs change time |
| INCREMENTAL_LEVEL | NUMBER | Normal full backups have a NULL value, level 0 incremental backups have a value of 0, and level 1 incremental backups have a value of 1 |
| INCREMENTAL_CHANGE# | NUMBER | Incremental change SCN |
| CHECKPOINT_CHANGE# | NUMBER | Checkpoint change SCN |
| CHECKPOINT_TIME | DATE | Checkpoint change time |
| MARKED_CORRUPT | NUMBER | Number of blocks marked corrupt |
| FILESIZE | NUMBER | File size, in bytes |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the datafile and the blocks that RMAN backed up. This is *not* the the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| TS# | NUMBER | Tablespace number |
| TSNAME | VARCHAR2(30) | Tablespace name |
| FILESIZE_DISPLAY | VARCHAR2(4000) | Same value as the FILESIZE column, but converted to a user-displayable format, for example *n*M, *n*G, *n*T, *n*P, and so on |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_DATAFILE_SUMMARY

V$BACKUP_DATAFILE_SUMMARY provides summary information for a specific criteria set, based on a backup job, a time range applicable to jobs, or a specific data file).

| Column | Datatype | Description |
|---|---|---|
| NUM_FILES_BACKED | NUMBER | Number of files backed up for specified criteria |
| NUM_DISTINCT_FILES_BACKED | NUMBER | Number of distinct files backed up |
| NUM_DISTINCT_TS_BACKED | NUMBER | Number of distinct tablespaces backed up |
| MIN_CHECKPOINT_CHANGE# | NUMBER | Minimum checkpoint change number of the data file for specified criteria |
| MAX_CHECKPOINT_CHANGE# | NUMBER | Maximum checkpoint change number of the data file for specified criteria |
| MIN_CHECKPOINT_TIME | DATE | Minimum checkpoint time of the data file for specified criteria |
| MAX_CHECKPOINT_TIME | DATE | Maximum checkpoint time of the data file for specified criteria |
| INPUT_BYTES | NUMBER | Total input bytes of files read |
| OUTPUT_BYTES | NUMBER | Total output bytes written |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the datafile and the blocks that RMAN backed up. This is *not* the the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for input bytes |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_DEVICE

V$BACKUP_DEVICE displays information about supported backup devices. If a device type does not support named devices, then one row with the device type and a null device name is returned for that device type. If a device type supports named devices then one row is returned for each available device of that type. The special device type DISK is not returned by this view because it is always available.

| Column | Datatype | Description |
|---|---|---|
| DEVICE_TYPE | VARCHAR2(17) | Type of the backup device |
| DEVICE_NAME | VARCHAR2(513) | Name of the backup device |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_FILES

V$BACKUP_FILES displays information about all RMAN backups (both image copies and backup sets) and archived logs.

This view simulates the LIST BACKUP and LIST COPY RMAN commands. This view requires that the database be set using the DBMS_RCVMAN.SETDATABASE procedure.

| Column | Datatype | Description |
|---|---|---|
| PKEY | NUMBER | Primary key for the backup |
| BACKUP_TYPE | VARCHAR2(32) | Type of the backup:<br>• BACKUP SET<br>• COPY<br>• PROXY COPY |
| FILE_TYPE | VARCHAR2(32) | Type of the file:<br>• DATAFILE<br>• CONTROLFILE<br>• SPFILE<br>• REDO LOG<br>• ARCHIVED LOG<br>• COPY (for an image copy backup)<br>• PIECE (for a backup piece) |
| KEEP | VARCHAR2(3) | Indicates whether the backup has a retention policy different from the value for CONFIGURE RETENTION POLICY (YES) or not (NO) |
| KEEP_UNTIL | DATE | If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the date after which the backup becomes obsolete. If the column is null and KEEP_OPTIONS is not null, the backup never becomes obsolete. |
| KEEP_OPTIONS | VARCHAR2(13) | KEEP options for the backup:<br>• LOGS - RMAN keeps the logs needed to recover the backup<br>• NOLOGS - RMAN does not keep the logs needed to recover the backup<br>If this column is null, then the backup has no KEEP options and will be made obsolete based on the retention policy. |
| STATUS | VARCHAR2(16) | Status of the backup: |

| Column | Datatype | Description |
|---|---|---|
| | | • AVAILABLE<br>• UNAVAILABLE<br>• EXPIRED<br>• OTHER |
| FNAME | VARCHAR2(1024) | Name of the file |
| TAG | VARCHAR2(32) | Tag of the piece, copy, or proxy copy |
| MEDIA | VARCHAR2(80) | Media ID of the piece or proxy copy |
| RECID | NUMBER | Recid of the record in the control file |
| STAMP | NUMBER | Stamp of the record in the control file |
| DEVICE_TYPE | VARCHAR2(255) | Type of media device that stores the backup |
| BLOCK_SIZE | NUMBER | Block size for the backup (in bytes) |
| COMPLETION_TIME | DATE | Time when the backup completed |
| COMPRESSED | VARCHAR2(3) | Indicates whether the backup piece is compressed (YES) or not (NO); valid only if FILE_TYPE is PIECE. Image copies cannot be compressed. |
| OBSOLETE | VARCHAR2(3) | Indicates whether the backup piece or copy is obsolete (YES) or not (NO); valid only if FILE_TYPE is PIECE or COPY |
| BYTES | NUMBER | Size of the file (in bytes) |
| BS_KEY | NUMBER | Primary key of the backup set (valid only if BACKUP_TYPE is BACKUP SET) |
| BS_COUNT | NUMBER | Count of the backup set from the control file record (valid only if BACKUP_TYPE is BACKUP SET) |
| BS_STAMP | NUMBER | Stamp of the backup set from the control file record (valid only if BACKUP_TYPE is BACKUP SET) |
| BS_TYPE | VARCHAR2(32) | Type of the backup set (valid only if BACKUP_TYPE is BACKUP SET):<br>• DATAFILE<br>• ARCHIVED LOG |
| BS_INCR_TYPE | VARCHAR2(32) | Incremental type of the backup set (valid only if BACKUP_TYPE is BACKUP SET) |
| BS_PIECES | NUMBER | Number of backup pieces in the backup set (valid only if BACKUP_TYPE is BACKUP SET) |
| BS_COPIES | NUMBER | Number of copies of the backup set (valid only if FILE_TYPE is PIECE and BACKUP_TYPE is BACKUP SET) |
| BS_COMPLETION_TIME | DATE | Completion time of the backup set (valid only if BACKUP_TYPE is BACKUP SET) |
| BS_STATUS | VARCHAR2(16) | Status of the backup set (valid only if BACKUP_TYPE is BACKUP SET):<br>• AVAILABLE |

| Column | Datatype | Description |
|---|---|---|
| | | • UNAVAILABLE<br>• EXPIRED<br>• OTHER - Pieces which are part of the backup set do not have uniform status (that is, some of them are available, some not) |
| BS_BYTES | NUMBER | Sum of all backup piece sizes in the backup set (valid only if BACKUP_TYPE is BACKUP SET) |
| BS_COMPRESSED | VARCHAR2(3) | Indicates whether the backup pieces of the backup set are compressed (YES) or not (NO); valid only if BACKUP_TYPE is BACKUP SET |
| BS_TAG | VARCHAR2(1024) | Tags of the backup set. If pieces have different tags, then all piece tags are concatenated and separated by commas. Valid only if BACKUP_TYPE is BACKUP SET) |
| BS_DEVICE_TYPE | VARCHAR2(255) | Device type of the backup set. If there is more than one device type, then they are separated by commas. Valid only if BACKUP_TYPE is BACKUP SET |
| BP_PIECE# | NUMBER | Number of pieces inside the backup set (valid only if FILE_TYPE is PIECE and BACKUP_TYPE is BACKUP SET) |
| BP_COPY# | NUMBER | Number of copies of the backup set (valid only if FILE_TYPE is PIECE and BACKUP_TYPE is BACKUP SET) |
| DF_FILE# | NUMBER | Absolute file number of the data file (valid only if FILE_TYPE is DATAFILE) |
| DF_TABLESPACE | VARCHAR2(30) | Tablespace name of the data file (valid only if FILE_TYPE is DATAFILE) |
| DF_RESETLOGS_CHANGE# | NUMBER | System change number (SCN) of the most recent RESETLOGS when the control file or data file was created (valid only if FILE_TYPE is DATAFILE) |
| DF_CREATION_CHANGE# | NUMBER | Creation SCN of the control file or data file (valid only if FILE_TYPE is CONTROLFILE or DATAFILE) |
| DF_CHECKPOINT_CHANGE# | NUMBER | System change number (SCN) of the most recent control file or data file checkpoint (valid only if FILE_TYPE is CONTROLFILE or DATAFILE) |
| DF_CKP_MOD_TIME | DATE | Modification time in case of SPFILE, otherwise time when the control file or data file was checkpointed (valid only if FILE_TYPE is SPFILE, CONTROLFILE, or DATAFILE) |
| RL_THREAD# | NUMBER | Redo log thread number of the archived log (valid only if FILE_TYPE is REDO LOG) |
| RL_SEQUENCE# | NUMBER | Redo log sequence number of the archived log (valid only if FILE_TYPE is REDO LOG) |
| RL_RESETLOGS_CHANGE# | NUMBER | System change number (SCN) of the most recent RESETLOGS when the record was created (valid only if FILE_TYPE is REDO LOG) |
| RL_FIRST_CHANGE# | NUMBER | First SCN of the redo log (valid only if FILE_TYPE is REDO LOG) |

| Column | Datatype | Description |
|---|---|---|
| RL_FIRST_TIME | DATE | Time when the Oracle Database switched into the redo log (valid only if FILE_TYPE is REDO LOG) |
| RL_NEXT_CHANGE# | NUMBER | First SCN of the next redo log in the thread (valid only if FILE_TYPE is REDO LOG) |
| RL_NEXT_TIME | DATE | First timestamp of the next redo log in the thread (valid only if FILE_TYPE is REDO LOG) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Backup and Recovery User's Guide* for more information about the DBMS_RCVMAN.SETDATABASE procedure

# V$BACKUP_NONLOGGED

V$BACKUP_NONLOGGED displays information about nonlogged block ranges in data file backups recorded in the control file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Nologged backup record ID |
| STAMP | NUMBER | Nonlogged backup record stamp |
| SET_STAMP | NUMBER | Backup set stamp |
| SET_COUNT | NUMBER | Backup set count |
| PIECE# | NUMBER | Backup piece that contains the range of nonlogged blocks |
| FILE# | NUMBER | Absolute file number of the data file that contains this range of nonlogged blocks |
| BLOCK# | NUMBER | Block number of the first nonlogged block in the range of nologged blocks |
| BLOCKS | NUMBER | Number of nonlogged blocks found starting with BLOCK# |
| NONLOGGED_CHANGE# | NUMBER | The smallest SCN on which any block in this block range became nonlogged. NULL if unknown. |
| NONLOGGED_TIME | VARCHAR2 | The time that corresponds to NONLOGGED_CHANGE#. NULL if unknown. |
| RESETLOGS_CHANGE# | VARCHAR2 | The resetlogs SCN of the incarnation on which this block range was first marked as nonlogged. NULL if unknown. |

| Column | Datatype | Description |
|---|---|---|
| RESETLOGS_TIME | VARCHAR2 | The resetlogs time of the incarnation on which this block range was first marked as nologged. NULL if unknown. |
| OBJECT# | VARCHAR2 | The object ID this range belongs to. If this field is NULL, the object number is unknown. |
| REASON | CHAR(7) | The reason why this block range appears in this list. For Oracle Database 12*c*, it is always UNKNOWN. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_PIECE

V$BACKUP_PIECE displays information about backup pieces from the control file. Each backup set consists of one or more backup pieces.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Backup piece record ID |
| STAMP | NUMBER | Backup piece record stamp |
| SET_STAMP | NUMBER | Backup set stamp |
| SET_COUNT | NUMBER | Backup set count |
| PIECE# | NUMBER | Backup piece number (1-N) |
| COPY# | NUMBER | Indicates the copy number for backup pieces created with duplex enabled. 1 if the backup piece is not duplexed. |
| DEVICE_TYPE | VARCHAR2(17) | Type of the device on which the backup piece resides. Set to DISK for backup sets on disk.<br>**See Also:** V$BACKUP_DEVICE |
| HANDLE | VARCHAR2(513) | Backup piece handle identifies the backup piece on restore |
| COMMENTS | VARCHAR2(64) | Comment returned by the operating system or storage subsystem. Set to NULL for backup pieces on disk. This value is informational only; not needed for restore. |
| MEDIA | VARCHAR2(65) | Name of the media on which the backup piece resides. This value is informational only; not needed for restore. |
| MEDIA_POOL | NUMBER | The media pool in which the copy resides. This is the same value that was entered in the POOL operand of the Recovery Manager BACKUP command. |
| CONCUR | VARCHAR2(3) | (YES \| NO) Indicates whether the piece on a media that can be accessed concurrently |

| Column | Datatype | Description |
|---|---|---|
| TAG | VARCHAR2(32) | Backup piece tag. The tag is specified at backup set level, but stored at piece level. |
| STATUS | VARCHAR2(1) | Indicates the status of the piece: A (available), D (deleted), or X (expired) |
| START_TIME | DATE | Starting time |
| COMPLETION_TIME | DATE | Completion time |
| ELAPSED_SECONDS | NUMBER | Number of elapsed seconds |
| DELETED | VARCHAR2(3) | (YES/NO) NO indicates that the file still exists. YES indicates the file no longer exists because it has been deleted. |
| BYTES | NUMBER | Size of the backup piece (in bytes) |
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether the file was created in the fast recovery area (YES) or not (NO) |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS record stamp |
| COMPRESSED | VARCHAR2(3) | Indicates whether the backup piece is compressed (YES) or not (NO) |
| BACKED_BY_VSS | VARCHAR2(3) | Whether or not the file has been backed up by Volume Shadow Copy Service (VSS). This column is reserved for internal use. |
| ENCRYPTED | VARCHAR2(3) | A value of YES means an encrypted backup, otherwise not an encrypted backup. |
| BACKED_BY_OSB | VARCHAR2(3) | A value of YES means the backup was done to Oracle Secure Backup. Otherwise, backed up by other third party tape library. |
| FOR_XTTS | VARCHAR2(3) | Indicates whether this is a cross platform backup piece:<br>• YES: This is a cross platform backup piece.<br>• NO: This is not a cross platform backup piece. |
| SAME_ENDIAN | VARCHAR2(3) | If the value of FOR_XTTS is YES, then this column indicates whether the backup piece has the same endianess as the current database (YES) or not (NO); otherwise NULL. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_PIECE_DETAILS

V$BACKUP_PIECE_DETAILS displays information about all available backup pieces.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session recid |
| SESSION_STAMP | NUMBER | Session stamp |
| BS_KEY | NUMBER | Backup set identifier |
| BP_KEY | NUMBER | Backup piece key |
| RECID | NUMBER | Backup piece record ID |
| STAMP | NUMBER | Backup piece record stamp |
| SET_STAMP | NUMBER | Backup set stamp |
| SET_COUNT | NUMBER | Backup set count |
| PIECE# | NUMBER | Backup piece number (1-N) |
| COPY# | NUMBER | Indicates the copy number for backup pieces created with duplex enabled. The value is 1 if the backup piece is not duplexed. |
| DEVICE_TYPE | VARCHAR2(17) | Type of device on which the backup piece resides. Set to DISK for backup sets on disk. |
| HANDLE | VARCHAR2(513) | Backup piece handle identifies the backup piece on restore |
| COMMENTS | VARCHAR2(64) | Comments returned by the operating system or storage subsystem. Set to NULL for backup pieces on disk. This value is informational only. It is not needed for restore. |
| MEDIA | VARCHAR2(65) | Name of the media on which the backup piece resides. This value is informational only. It is not needed for restore. |
| MEDIA_POOL | NUMBER | The media pool in which the copy resides. This is the same value that was entered in the POOL operand of the Recovery Manager BACKUP command. |
| CONCUR | VARCHAR2(3) | (YES\|NO) indicates whether or not the piece is on a media that can be accessed concurrently |
| TAG | VARCHAR2(32) | Backup piece tag. The tag is specified at backup set level, but stored at piece level. |
| STATUS | VARCHAR2(1) | Indicates the status of the piece: A (available), D (deleted), or X (expired) |
| START_TIME | DATE | Starting time |
| COMPLETION_TIME | DATE | Completion time |
| ELAPSED_SECONDS | NUMBER | Number of elapsed seconds |
| DELETED | VARCHAR2(3) | NO indicates that the file still exists. YES indicates that the file no longer exists because it has been deleted. |
| BYTES | NUMBER | Size of the backup piece, in bytes |

| Column | Datatype | Description |
|---|---|---|
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether or not the file was created in the fast recovery area (YES) or not (NO) |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS record stamp |
| COMPRESSED | VARCHAR2(3) | Indicates whether the backup piece is compressed (YES) or not (NO) |
| BACKED_BY_VSS | VARCHAR2(3) | Whether or not the file has been backed up by Volume Shadow Copy Service (VSS). This column is reserved for internal use. |
| ENCRYPTED | VARCHAR2(3) | A value of YES means an encrypted backup, otherwise not an encrypted backup. |
| BACKED_BY_OSB | VARCHAR2(3) | A value of YES means the backup was done to Oracle Secure Backup. Otherwise, backed up by other third party tape library. |
| FOR_XTTS | VARCHAR2(3) | Indicates whether this is a cross platform backup piece:<br>• YES: This is a cross platform backup piece.<br>• NO: This is not a cross platform backup piece. |
| SAME_ENDIAN | VARCHAR2(3) | If the value of FOR_XTTS is YES, then this column indicates whether the backup piece has the same endianess as the current database (YES) or not (NO); otherwise NULL. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| PIECES_PER_SET | NUMBER | Number of backup pieces per set |
| SIZE_BYTES_DISPLAY | VARCHAR2(4000) | Size (in bytes) of the backup piece to display |

# V$BACKUP_REDOLOG

V$BACKUP_REDOLOG displays information about archived logs in backup sets from the control file. Note that online redo logs cannot be backed up directly; they must be archived first to disk and then backed up. An archive log backup set can contain one or more archived logs.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Record ID for this row; it is an integer that identifies this row |
| STAMP | NUMBER | Timestamp used with RECID to uniquely identify this row |
| SET_STAMP | NUMBER | One of the foreign keys for the row of the V$BACKUP_SET table that identifies this backup set |

| Column | Datatype | Description |
|---|---|---|
| SET_COUNT | NUMBER | One of the foreign keys for the row of the V$BACKUP_SET table that identifies this backup set |
| THREAD# | NUMBER | Thread number for the log |
| SEQUENCE# | NUMBER | Log sequence number |
| RESETLOGS_CHANGE# | NUMBER | Change number of the last resetlogs before the log was written |
| RESETLOGS_TIME | DATE | Change time of the last resetlogs before the log was written. These will be the same for all logs in a backup set. |
| FIRST_CHANGE# | NUMBER | SCN when the log was switched into. The redo in the log is at this SCN and greater. |
| FIRST_TIME | DATE | Time allocated when the log was switched into |
| NEXT_CHANGE# | NUMBER | SCN when the next log in this thread was switched into. The redo in the log is below this SCN. |
| NEXT_TIME | DATE | Time when the next log in this thread was switched into |
| BLOCKS | NUMBER | Size of the log in logical blocks including the header block |
| BLOCK_SIZE | NUMBER | Size of the log blocks in bytes |
| TERMINAL | VARCHAR2(3) | Indicates whether this record corresponds to a terminal archived redo log, as defined in V$ARCHIVED_LOG (YES) or not (NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_SET

V$BACKUP_SET displays information about backup sets from the control file. A backup set record is inserted after the backup set is successfully completed.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Backup set record ID |
| STAMP | NUMBER | Backup set record stamp |
| SET_STAMP | NUMBER | Backup set stamp. The backup set stamp and count uniquely identify the backup set.<br>Primary key for the V$BACKUP_SET table, and the foreign key for the following tables: V$BACKUP_PIECE, V$BACKUP_DATAFILE, V$BACKUP_REDOLOG, V$BACKUP_CORRUPTION. |
| SET_COUNT | NUMBER | Backup set count. The backup set count is incremented by one every time a new backup set is started (if the backup set is never |

| Column | Datatype | Description |
|---|---|---|
| | | completed the number is "lost"). If the control file is re-created then the count is reset to 1. Therefore the count must be used with the stamp to uniquely identify a backup set.<br>Primary key for the V$BACKUP_SET table, and the foreign key for the following tables: V$BACKUP_PIECE, V$BACKUP_DATAFILE, V$BACKUP_REDOLOG, V$BACKUP_CORRUPTION |
| BACKUP_TYPE | VARCHAR2(1) | Type of files that are in this backup. If the backup contains archived redo logs, the value is L. If this is a data file full backup, the value is D. If this is an incremental backup, the value is I. |
| CONTROLFILE_INCLUDED | VARCHAR2(3) | Set to YES if there is a control file included in this backup set, otherwise set to NO |
| INCREMENTAL_LEVEL | NUMBER | Location where this backup set fits into the database's backup strategy. Set to NULL for full datafile, archivelog, controlfile, and spfile backups, set to 0 for incremental level 0 datafile backups, and set to 1 for incremental level 1 datafile backups. |
| PIECES | NUMBER | Number of distinct backup pieces in the backup set |
| START_TIME | DATE | Starting time |
| COMPLETION_TIME | DATE | Time that this backup set completed |
| ELAPSED_SECONDS | NUMBER | The number of elapsed seconds |
| BLOCK_SIZE | NUMBER | Block size of the backup set |
| INPUT_FILE_SCAN_ONLY | VARCHAR2(3) | YES indicates no actual backup is performed, but the data files are read. NO indicates a normal backup is performed. |
| KEEP | VARCHAR2(3) | (YES/NO) Indicates whether or not this backup set has a retention policy that is different than the value for the configure retention policy |
| KEEP_UNTIL | DATE | If KEEP_UNTIL_TIME is specified, this is the date after which the backup becomes obsolete. If this column is null, then the backup never expires. |
| KEEP_OPTIONS | VARCHAR2(11) | Lists additional retention options for this backup set. Possible values are:<br>LOGS - The logs need to recover this backup are kept<br>NOLOGS - The logs needed to recover this backup will not be kept<br>BACKUP_LOGS - An archive log backup exists to support this backup set |
| MULTI_SECTION | VARCHAR2(3) | Indicates whether or not this backup set is a multi-section backup. Valid values are YES and NO. A multi-section backup is a backup in which multiple backup pieces are produced independently in parallel by multiple channels. |
| FOR_XTTS | VARCHAR2(3) | Indicates whether this is a cross platform backup set:<br>• YES: This is a cross platform backup set.<br>• NO: This is not a cross platform backup set. |

| Column | Datatype | Description |
|---|---|---|
| SAME_ENDIAN | VARCHAR2(3) | If the value of FOR_XTTS is YES, then this column indicates whether the backup set has the same endianess as the current database (YES) or not (NO); otherwise NULL. |
| INC_DMPFILE | VARCHAR2(3) | If the value of FOR_XTTS is YES, then this column indicates whether the backup set includes a Data Pump export file of the backed up data files (YES) or not (NO), otherwise NULL. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_SET_DETAILS

V$BACKUP_SET_DETAILS provides detailed information about the backup set. This view will contain an extra row for each backup session that invokes BACKUP BACKUPSET (that is, creates new copies for the same backup set or copies backup set information from disk to tape). However, the remaining values of other columns belong to the complete backup set.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Identifies the job, together with SESSION_STAMP. |
| SESSION_STAMP | NUMBER | Identifies the job, together with SESSION_RECID |
| BS_KEY | NUMBER | Backup set identifier |
| RECID | NUMBER | RECID from V$BACKUP_SET record |
| STAMP | NUMBER | Stamp from V$BACKUP_SET record |
| SET_STAMP | NUMBER | Backup set stamp |
| SET_COUNT | NUMBER | Backup count number |
| BACKUP_TYPE | VARCHAR2(1) | Type of backup (same as in V$BACKUP_SET) |
| CONTROLFILE_INCLUDED | VARCHAR2(3) | Control file included in backup set (same as in V$BACKUP_SET) |
| INCREMENTAL_LEVEL | NUMBER | Incremental level (same as in V$BACKUP_SET) |
| PIECES | NUMBER | Number of pieces (same as in V$BACKUP_SET) |
| START_TIME | DATE | Start time of the backup set (same as in V$BACKUP_SET) |
| COMPLETION_TIME | DATE | Completion time of the backup set (same as in V$BACKUP_SET) |
| ELAPSED_SECONDS | NUMBER | Time taken for backup set creation (same as in V$BACKUP_SET) |

| Column | Datatype | Description |
|---|---|---|
| BLOCK_SIZE | NUMBER | Block size (same as in V$BACKUP_SET) |
| KEEP | VARCHAR2(3) | Keep value (same as in V$BACKUP_SET) |
| KEEP_UNTIL | DATE | Keep Until time (same as in V$BACKUP_SET) |
| KEEP_OPTIONS | VARCHAR2(11) | Keep options (same as in V$BACKUP_SET) |
| DEVICE_TYPE | VARCHAR2(17) | Type of device. If the backup set exists on more than one device type, an * is indicated here. |
| COMPRESSED | VARCHAR2(3) | YES, if backup is compressed |
| NUM_COPIES | NUMBER | Number of identical copies |
| OUTPUT_BYTES | NUMBER | Size of the backup set |
| ORIGINAL_INPUT_BYTES | NUMBER | Amount of data backed up when the backup set was created |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the datafile and the blocks that RMAN backed up. This is *not* the the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| STATUS | CHAR(1) | The status of the backup set. It is always A (all backup pieces available), because this view only reflects available backup sets. |
| ORIGINAL_INPRATE_BYTES | NUMBER | Number of bytes read per second when backup set was initially created |
| OUTPUT_RATE_BYTES | NUMBER | Number of bytes written per second when the backup set was initially created |
| ORIGINAL_INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Input rate to display |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Size of backup set to display |
| ORIGINAL_INPRATE_BYTES_DISPLAY | VARCHAR2(4000) | Input rate per second for display |
| OUTPUT_RATE_BYTES_DISPLAY | VARCHAR2(4000) | Output rate per second for display |
| TIME_TAKEN_DISPLAY | VARCHAR2(4000) | Elapsed time in hh:mm:ss format |
| ENCRYPTED | VARCHAR2(3) | A value of YES means an encrypted backup, otherwise not an encrypted backup. |
| BACKED_BY_OSB | VARCHAR2(3) | A value of YES means the backup was done to Oracle Secure Backup. Otherwise, backed up by other third party tape library. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$BACKUP_SET"

# V$BACKUP_SET_SUMMARY

V$BACKUP_SET_SUMMARY provides summary information for a backup set.

| Column | Datatype | Description |
|---|---|---|
| NUM_BACKUPSETS | NUMBER | Total number of backup sets created |
| OLDEST_BACKUP_TIME | DATE | Oldest backup start time |
| NEWEST_BACKUP_TIME | DATE | Newest backup start time |
| OUTPUT_BYTES | NUMBER | Number of output bytes (not including multiple copies) |
| ORIGINAL_INPUT_BYTES | NUMBER | Number of input bytes when backup sets were created |
| ORIGINAL_INPRATE_BYTES | NUMBER | Average input rate |
| OUTPUT_RATE_BYTES | NUMBER | Average output rate |
| COMPRESSION_RATIO | NUMBER | The ratio between the total blocks in the datafile and the blocks that RMAN backed up. This is *not* the the ratio from the AS COMPRESSED BACKUPSET clause of the BACKUP command. |
| ORIGINAL_INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for input bytes |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| ORIGINAL_INPRATE_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for input rate |
| OUTPUT_RATE_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output rate |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_SPFILE

V$BACKUP_SPFILE displays information about server parameter files in backup sets from the control file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Backup SPFILE record ID |
| STAMP | NUMBER | Backup SPFILE record stamp |
| SET_STAMP | NUMBER | Backup set stamp (of the set which contains this SPFILE backup) |
| SET_COUNT | NUMBER | Backup set count (of the set which contains this SPFILE backup) |
| MODIFICATION_TIME | DATE | Time when the SPFILE was last modified (this also includes creation time) |
| BYTES | NUMBER | Size of the SPFILE (in bytes) |
| COMPLETION_TIME | DATE | Time when the backup of the SPFILE completed |
| DB_UNIQUE_NAME | VARCHAR2(30) | Unique database name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_SPFILE_DETAILS

V$BACKUP_SPFILE_DETAILS displays information about all restorable SP files backed up in the backup set.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session recid |
| SESSION_STAMP | NUMBER | Session stamp |
| BS_KEY | NUMBER | Unique backup set identifier |
| SET_STAMP | NUMBER | Backup set stamp |
| SET_COUNT | NUMBER | Backup set count |
| MODIFICATION_TIME | DATE | Time the backup set was modified |
| FILESIZE | NUMBER | Size, in bytes, of the SPFILE that was backed up |
| FILESIZE_DISPLAY | VARCHAR2(4000) | Same value as the FILESIZE column, but converted to a user-displayable format, for example *n*M, *n*G, *n*T, *n*P, and so on |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_SPFILE_SUMMARY

V$BACKUP_SPFILE_SUMMARY provides summary information for input SP file, based on either a backup job or time range applicable to jobs.

| Column | Datatype | Description |
|---|---|---|
| NUM_FILES_BACKED | NUMBER | Number of files backed up |
| NUM_DISTINCT_FILES_BACKED | NUMBER | Number of distinct SP files backed up (with modification timestamp) |
| MIN_MODIFICATION_TIME | DATE | Minimum modification time |
| MAX_MODIFICATION_TIME | DATE | Maximum modification time |
| INPUT_BYTES | NUMBER | Total input bytes for all SP files backed up |
| INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for all input bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BACKUP_SYNC_IO

V$BACKUP_SYNC_IO displays performance information about ongoing and recently completed RMAN backups and restores. For each backup, it contains one row for each input data file, one row for the aggregate total performance of all data files, and one row for the output backup piece. This data is not stored persistently, and is not preserved when the instance is re-started.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | The Oracle SID of the session doing the backup or restore |
| SERIAL | NUMBER | The use count for the SID doing the backup or restore |

| Column | Datatype | Description |
|---|---|---|
| USE_COUNT | NUMBER | A counter that can be used to identify rows from different backup sets |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS record stamp |
| DEVICE_TYPE | VARCHAR2(17) | The device type where the file is located |
| TYPE | VARCHAR2(9) | INPUT, OUTPUT, or AGGREGATE |
| STATUS | VARCHAR2(11) | NOT STARTED, IN PROGRESS, or FINISHED |
| FILENAME | VARCHAR2(513) | The name of the backup file being read or written |
| SET_COUNT | NUMBER | The set count of the backup set being read or written |
| SET_STAMP | NUMBER | The set stamp of the backup set being read or written |
| BUFFER_SIZE | NUMBER | The size of the buffers being used to read/write this file, in bytes |
| BUFFER_COUNT | NUMBER | The number of buffers being used to read/write this file |
| TOTAL_BYTES | NUMBER | The total number of bytes that will be read or written for this file, if known. If not known, this column will be null. |
| OPEN_TIME | DATE | The time this file was opened. If TYPE='AGGREGATE', then this is the time that the first file in the aggregate was opened. |
| CLOSE_TIME | DATE | The time this file was closed. If TYPE='AGGREGATE', then this is the time that the last file in the aggregate was closed. |
| ELAPSED_TIME | NUMBER | The time, in hundredths of a second, that the file was open |
| MAXOPENFILES | NUMBER | The number of concurrently open DISK files. This value is only present in rows where TYPE='AGGREGATE'. |
| BYTES | NUMBER | The number of bytes read or written so far |
| EFFECTIVE_BYTES_PER_SECOND | NUMBER | The I/O rate that was achieved with this device during this backup |
| IO_COUNT | NUMBER | The number of I/Os that were performed to this file |
| IO_TIME_TOTAL | NUMBER | The total time, in hundredths of a second, taken to do I/O for this file |
| IO_TIME_MAX | NUMBER | The maximum time taken for a single I/O request |
| DISCRETE_BYTES_PER_SECOND | NUMBER | The average transfer rate for this file |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BGPROCESS

V$BGPROCESS displays information about the background processes.

| Column | Datatype | Description |
|---|---|---|
| PADDR | RAW(4 \| 8) | Address of the process state object |
| PSERIAL# | NUMBER | Process state object serial number |
| NAME | VARCHAR2(5) | Name of this background process |
| DESCRIPTION | VARCHAR2(64) | Description of the background process |
| ERROR | NUMBER | Error encountered |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BH

V$BH displays the status and number of pings for every buffer in the SGA. This is an Oracle Real Application Clusters view.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Data file identifier number (to find the file name, query DBA_DATA_FILES or V$DBFILE) |
| BLOCK# | NUMBER | Block number |
| CLASS# | NUMBER | Class number |
| STATUS | VARCHAR2(10) | Status of the buffer:<br>• free - Not currently in use<br>• xcur - Exclusive<br>• scur - Shared current<br>• cr - Consistent read<br>• read - Being read from disk<br>• mrec - In media recovery mode<br>• irec - In instance recovery mode<br>• pi - A past image in RAC mode<br>• securefile - A secured file buffer<br>• flashfree - A free flash cache buffer<br>• flashcur - A current flash cache buffer |
| XNC | NUMBER | This column is obsolete. Its value is hard-coded to 0. |
| FORCED_READS | NUMBER | This column is obsolete. Its value is hard-coded to 0. |
| FORCED_WRITES | NUMBER | This column is obsolete. Its value is hard-coded to 0. |

| Column | Datatype | Description |
|---|---|---|
| LOCK_ELEMENT_ADDR | RAW(4 \| 8) | Address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| LOCK_ELEMENT_NAME | NUMBER | The address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| LOCK_ELEMENT_CLASS | NUMBER | The address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| DIRTY | VARCHAR2(1) | Y - block modified |
| TEMP | VARCHAR2(1) | Y - temporary block |
| PING | VARCHAR2(1) | Y - block pinged |
| STALE | VARCHAR2(1) | Y - block is stale |
| DIRECT | VARCHAR2(1) | Y - direct block |
| NEW | CHAR(1) | Always set to N. This column is obsolete and maintained for backward compatibility. |
| OBJD | NUMBER | Database object number of the block that the buffer represents |
| TS# | NUMBER | Tablespace number of block |
| LOBID | NUMBER | If the buffer belongs to a SecureFiles object, the value in this column is the unique identifier for the SecureFiles object. For other buffer types, the value in this column is meaningless. |
| CACHEHINT | NUMBER | Numeric representation of the values in the FLASH_CACHE and CELL_FLASH_CACHE columns. See the descriptions and possible values for the FLASH_CACHE and CELL_FLASH_CACHE columns in this view. |
| FLASH_CACHE | VARCHAR2(7) | Database Smart Flash Cache hint to be used for segment blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>Solaris and Oracle Linux functionality only. |
| CELL_FLASH_CACHE | VARCHAR2(7) | Cell flash cache hint to be used for segment blocks:<br>• DEFAULT<br>• KEEP<br>• NONE<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BLOCK_CHANGE_TRACKING

V$BLOCK_CHANGE_TRACKING displays the status of block change tracking for the database.

| Column | Datatype | Description |
|---|---|---|
| STATUS | VARCHAR2(10) | Status of block change tracking in the database:<br>• DISABLED - Block change tracking is disabled<br>• TRANSITION - Block change tracking is in the process of transitioning between the enabled and disabled states. The TRANSITION state should usually never be observed, because it only exists while enabling or disabling block change tracking. This state might be observed if the instance crashed while enabling or disabling block change tracking, in which case it will be cleaned up automatically the next time that the database is opened.<br>• ENABLED - Block change tracking is enabled |
| FILENAME | VARCHAR2(513) | Name of the block change tracking file for the database |
| BYTES | NUMBER | Size of the block change tracking file (in bytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Backup and Recovery User's Guide* for information on setting up block change tracking

# V$BLOCKING_QUIESCE

V$BLOCKING_QUIESCE indicates if a session is blocking, or would block, a quiesce operation.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BT_SCAN_CACHE

V$BT_SCAN_CACHE shows the parameters and status of the big table cache section.

| Column | Datatype | Description |
|---|---|---|
| BT_CACHE_ALLOC | NUMBER | Current ratio of the big table cache section to the buffer cache |
| BT_CACHE_TARGET | NUMBER | Target ratio of the big table cache section to the buffer cache |
| OBJECT_COUNT | NUMBER | Number of objects tracked by the big table cache section |
| MEMORY_BUF_ALLOC | NUMBER | Number of memory buffers allocated by the big table cache section to objects |
| MIN_CACHED_TEMP | NUMBER | Minimum temperature of any object that is allowed to be cached by the big table cache section |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DB_BIG_TABLE_CACHE_PERCENT_TARGET" for more information about enabling the big table cache

# V$BT_SCAN_OBJ_TEMPS

V$BT_SCAN_OBJ_TEMPS shows the active objects currently tracked by the big table cache.

| Column | Datatype | Description |
|---|---|---|
| TS# | NUMBER | Tablespace number |
| DATAOBJ# | NUMBER | Data object number (objd) |
| SIZE_IN_BLKS | NUMBER | Size of the object being scanned on this instance, in blocks |

| Column | Datatype | Description |
|---|---|---|
| TEMPERATURE | NUMBER | Temperature of this object |
| POLICY | VARCHAR2(10) | Caching policy of this object. Possible values:<br>• MEM_ONLY: This object will be fully cached in memory.<br>• MEM_PART: This object will be partially cached in memory and some portion will remain on disk and will not be cached.<br>• DISK: this object will not be cached in memory or flash for the scan at all.<br>• INVALID: The caching policy is not valid. |
| CACHED_IN_MEM | NUMBER | The number of blocks that are cached/allocated in memory for this object |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DB_BIG_TABLE_CACHE_PERCENT_TARGET" for more information about enabling the big table cache

# V$BUFFER_POOL

V$BUFFER_POOL displays information about all buffer pools available for the instance.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Buffer pool identifier number |
| NAME | VARCHAR2(20) | Name of the buffer pool:<br>• DEFAULT<br>• KEEP<br>• RECYCLE<br>**Note:** Currently, KEEP and RECYCLE pools only exist for the standard block size. All nonstandard block size pools are DEFAULT. |
| BLOCK_SIZE | NUMBER | Block size (in bytes) for buffers in this pool. Possible values: the standard block size, the power of 2 nonstandard block sizes, 2048, 4096, 8192, 16384, 32768. |
| RESIZE_STATE | VARCHAR2(10) | Current state of the resize operation: |

| Column | Datatype | Description |
|---|---|---|
| | | STATIC - Not being resized<br>ALLOCATING - Memory is being allocated (can be canceled by the user)<br>ACTIVATING - New buffers are being created (user cannot cancel)<br>SHRINKING - Buffers are being deleted (can be canceled by the user) |
| CURRENT_SIZE | NUMBER | Present size of the sub-cache (in megabytes) |
| BUFFERS | NUMBER | Current instantaneous number of buffers |
| TARGET_SIZE | NUMBER | If a resize is in progress (state is not STATIC), records new target size (in megabytes). If the pool is STATIC, the value in this column is the same as the current size of the pool. |
| TARGET_BUFFERS | NUMBER | If a resize is in progress, records new target size in terms of buffers. Otherwise, the value in this column is the same as the current number of buffers. |
| PREV_SIZE | NUMBER | Previous buffer pool size. If the buffer pool has never been resized, the previous size is zero. |
| PREV_BUFFERS | NUMBER | Previous number of buffers in the buffer pool. Value is zero if the buffer pool has never been resized. |
| LO_BNUM | NUMBER | Obsolete column |
| HI_BNUM | NUMBER | Obsolete column |
| LO_SETID | NUMBER | Obsolete column |
| HI_SETID | NUMBER | Obsolete column |
| SET_COUNT | NUMBER | Obsolete column |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"DB_BLOCK_SIZE"

# V$BUFFER_POOL_STATISTICS

V$BUFFER_POOL_STATISTICS displays statistics about all buffer pools available for the instance.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Buffer pool identifier number |
| NAME | VARCHAR2(20) | Name of the buffer pool |
| BLOCK_SIZE | NUMBER | Block size (in bytes) for buffers in this pool. Possible values: the standard block size, the power of 2 nonstandard block sizes, 2048, 4096, 8192, 16384, 32768. |
| SET_MSIZE | NUMBER | Buffer pool maximum set size |
| CNUM_REPL | NUMBER | Number of buffers on replacement list |
| CNUM_WRITE | NUMBER | Number of buffers on write list |
| CNUM_SET | NUMBER | Number of buffers in set |
| BUF_GOT | NUMBER | Number of buffers gotten by the set |
| SUM_WRITE | NUMBER | Number of buffers written by the set |
| SUM_SCAN | NUMBER | Number of buffers scanned in the set |
| FREE_BUFFER_WAIT | NUMBER | Free buffer wait statistic |
| WRITE_COMPLETE_WAIT | NUMBER | Write complete wait statistic |
| BUFFER_BUSY_WAIT | NUMBER | Buffer busy wait statistic |
| FREE_BUFFER_INSPECTED | NUMBER | Free buffer inspected statistic |
| DIRTY_BUFFERS_INSPECTED | NUMBER | Dirty buffers inspected statistic |
| DB_BLOCK_CHANGE | NUMBER | Database blocks changed statistic |
| DB_BLOCK_GETS | NUMBER | Database blocks gotten statistic |
| CONSISTENT_GETS | NUMBER | Consistent gets statistic |
| PHYSICAL_READS | NUMBER | Physical reads statistic |
| PHYSICAL_WRITES | NUMBER | Physical writes statistic |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"DB_CACHE_SIZE"

# V$BUFFERED_PUBLISHERS

V$BUFFERED_PUBLISHERS displays information about all buffered publishers in the instance. There is one row per queue per sender. The values are reset to zero when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Identifier for the queue |
| QUEUE_SCHEMA | VARCHAR2(30) | Owner of the queue |
| QUEUE_NAME | VARCHAR2(30) | Name of the queue |
| SENDER_NAME | VARCHAR2(30) | Name of the agent enqueuing the message (the Streams name assigned for a capture process) |
| SENDER_ADDRESS | VARCHAR2(1024) | Queue name and database name of the source (last propagating) queue; database name is not specified if the source queue is in the local database |
| SENDER_PROTOCOL | NUMBER | Protocol used by the sender's address |
| NUM_MSGS | NUMBER | Current number of messages that have yet to be dequeued from the buffer queue |
| CNUM_MSGS | NUMBER | Cumulative total number of messages enqueued into the buffered queue since the database last started (comparable to V$STREAMS_CAPTURE.TOTAL_ENQUEUED_MESSAGES) |
| LAST_ENQUEUED_MSG | NUMBER | Most recently enqueued message identifier |
| UNBROWSED_MSGS | NUMBER | Number of messages that have been enqueued but not browsed |
| OVERSPILLED_MSGS | NUMBER | Number of messages that have been spilled but not browsed |
| MEMORY_USAGE | NUMBER | Percentage of the Streams pool that is being used (or 0 if there is no Streams pool) |
| ELAPSED_ENQUEUE_TIME | NUMBER | Total time spent in enqueue (in hundredths of a second) |
| ENQUEUE_CPU_TIME | NUMBER | Total CPU time for enqueue (in hundredths of a second) |
| LAST_ENQUEUE_TIME | TIMESTAMP(3) WITH TIME ZONE | Last message enqueue time |
| PUBLISHER_STATE | VARCHAR2(59) | State of the publisher:<br>• IN FLOW CONTROL: TOO MANY UNBROWSED MESSAGES<br>• IN FLOW CONTROL: OVERSPILLED MESSAGES<br>• IN FLOW CONTROL: INSUFFICIENT MEMORY AND UNBROWSED MESSAGES<br>• PUBLISHING MESSAGES - Normal |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BUFFERED_QUEUES

V$BUFFERED_QUEUES displays information about all buffered queues in the instance. There is one row per queue.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Identifier for the queue |
| QUEUE_SCHEMA | VARCHAR2(30) | Owner of the queue |
| QUEUE_NAME | VARCHAR2(30) | Name of the queue |
| STARTUP_TIME | DATE | Startup time |
| NUM_MSGS | NUMBER | Total number of messages currently in the buffered queue |
| SPILL_MSGS | NUMBER | Current number of overflow messages spilled to disk from the buffered queue |
| CNUM_MSGS | NUMBER | Cumulative total number of messages enqueued into the buffered queue since the database last started |
| CSPILL_MSGS | NUMBER | Cumulative total number of overflow messages spilled to disk from the buffered queue since the database last started |
| EXPIRED_MSGS | NUMBER | Number of expired messages |
| OLDEST_MSGID | RAW(16) | Message ID of the oldest message |
| OLDEST_MSG_ENQTM | TIMESTAMP(3) | Enqueue time of the oldest message |
| QUEUE_STATE | VARCHAR2(25) | Indicates whether the queue is in recovery mode (QUEUE IS IN RECOVERY MODE) or not (NORMAL) |
| ELAPSED_ENQUEUE_TIME | NUMBER | Total time spent in enqueue (in hundredths of a second) |
| ELAPSED_DEQUEUE_TIME | NUMBER | Total time spent in dequeue (in hundredths of a second) |
| ELAPSED_TRANSFORMATION_TIME | NUMBER | Total time for evaluating transformations (in hundredths of a second) |
| ELAPSED_RULE_EVALUATION_TIME | NUMBER | Total time for rule evaluations (in hundredths of a second) |
| ENQUEUE_CPU_TIME | NUMBER | Total CPU time for enqueue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | Total CPU time for dequeue (in hundredths of a second) |
| AVG_MSG_AGE | NUMBER | Average age of messages in the queue |

| Column | Datatype | Description |
|---|---|---|
| LAST_ENQUEUE_TIME | TIMESTAMP(3) WITH TIME ZONE | Last message enqueue time |
| LAST_DEQUEUE_TIME | TIMESTAMP(3) WITH TIME ZONE | Last message dequeue time |
| QUEUE_SIZE | NUMBER | Size of queue, which is the total number of bytes allocated for all messages and metadata |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$BUFFERED_SUBSCRIBERS

V$BUFFERED_SUBSCRIBERS displays information about the subscribers for all buffered queues in the instance. There is one row per subscriber per queue.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Identifier for the queue |
| QUEUE_SCHEMA | VARCHAR2(30) | Owner of the queue |
| QUEUE_NAME | VARCHAR2(30) | Name of the queue |
| SUBSCRIBER_ID | NUMBER | Internal subscriber number (for identification) |
| SUBSCRIBER_NAME | VARCHAR2(30) | Name of the subscriber (comparable to the Streams name for the Streams process) |
| SUBSCRIBER_ADDRESS | VARCHAR2(1024) | Address of the subscribing agent |
| PROTOCOL | NUMBER | Protocol of the subscribing agent |
| SUBSCRIBER_TYPE | VARCHAR2(30) | Type of the subscriber:<br>• PROXY - Proxy subscriber<br>• SUBSCRIBER |
| STARTUP_TIME | DATE | Startup time |
| LAST_BROWSED_SEQ | NUMBER | Sequence number of the most recently browsed message for the subscriber (comparable to the number of messages in the V$STREAMS_APPLY_READER view) |
| LAST_BROWSED_NUM | NUMBER | Internal Message number for the most recently browsed message for the subscriber |

| Column | Datatype | Description |
|---|---|---|
| LAST_DEQUEUED_SEQ | NUMBER | Sequence number of the most recently dequeued message for the subscriber (comparable to the number of messages in the V$STREAMS_APPLY_COORDINATOR view) |
| LAST_DEQUEUED_NUM | NUMBER | Internal Message number for the most recently dequeued message for the subscriber |
| CURRENT_ENQ_SEQ | NUMBER | Current sequence number of the most recently enqueued message for the subscriber |
| NUM_MSGS | NUMBER | Total number of outstanding messages currently enqueued in the buffered queue for the subscriber (includes the count of the messages overflowed to disk) |
| CNUM_MSGS | NUMBER | Cumulative total number of messages enqueued for the subscriber since the database last started |
| TOTAL_DEQUEUED_MSG | NUMBER | Total number of messages dequeued by the subscriber |
| TOTAL_SPILLED_MSG | NUMBER | Total number of spilled messages for the subscriber |
| EXPIRED_MSGS | NUMBER | Number of expired messages |
| MESSAGE_LAG | NUMBER | Message lag of the subscriber |
| ELAPSED_DEQUEUE_TIME | NUMBER | Total time spent in dequeue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | Total CPU time for dequeue (in hundredths of a second) |
| AVG_MSG_AGE | NUMBER | Average age of messages currently enqueued in the buffered queue for the subscriber |
| LAST_DEQUEUE_TIME | TIMESTAMP(3) WITH TIME ZONE | Last message dequeue time |
| OLDEST_MSGID | RAW(16) | Message ID of the oldest message |
| OLDEST_MSG_ENQTM | TIMESTAMP(3) | Enqueue time of the oldest message |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CACHE

V$CACHE displays information from the block header of each block in the SGA of the current instance as related to particular database objects. This is an Oracle Real Application Clusters view.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Data file identifier number (to find the file name, query DBA_DATA_FILES or V$DBFILE) |
| BLOCK# | NUMBER | Block number |
| CLASS# | NUMBER | Class number |
| STATUS | VARCHAR2(10) | Status of the block:<br>• free - Not currently in use<br>• xcur - Exclusive<br>• scur - Shared current<br>• cr - Consistent read<br>• read - Being read from disk<br>• mrec - In media recovery mode<br>• irec - In instance recovery mode |
| XNC | NUMBER | Number of PCM x to null lock conversions due to contention with another instance. This column is obsolete and maintained for backward compatibility. |
| FORCED_READS | NUMBER | Number of times the block had to be reread from the cache because another instance has forced it out of this instance's cache by requesting the lock on the block in exclusive mode |
| FORCED_WRITES | NUMBER | Number of times GCS had to write this block to cache because this instance had used the block and another instance had requested the lock on the block in a conflicting mode |
| NAME | VARCHAR2(128) | Name of the database object containing the block |
| PARTITION_NAME | VARCHAR2(128) | Name of the partition (null for nonpartitioned objects) |
| KIND | VARCHAR2(15) | Type of the database object:<br>• INDEX<br>• TABLE<br>• CLUSTER<br>• VIEW<br>• SYNONYM<br>• SEQUENCE<br>• PROCEDURE<br>• FUNCTION<br>• PACKAGE<br>• NON-EXISTENT<br>• PACKAGE BODY<br>• TRIGGER<br>• TYPE<br>• TYPE BODY<br>• TABLE PARTITION<br>• INDEX PARTITION<br>• LOB<br>• LIBRARY<br>• JAVA SOURCE<br>• JAVA CLASS<br>• JAVA RESOURCE<br>• JAVA DATA |

| Column | Datatype | Description |
|---|---|---|
| | | • UNDO |
| OWNER# | NUMBER | Owner number |
| LOCK_ELEMENT_ADDR | RAW(4 \| 8) | Address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| LOCK_ELEMENT_NAME | NUMBER | Name of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CACHE_LOCK

V$CACHE_LOCK is deprecated. The information that was provided in this view is now provided in the V$INSTANCE_CACHE_TRANSFER and V$SEGMENT_STATISTICS views.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Data file identifier number (to find file name, query DBA_DATA_FILES or V$DBFILE) |
| BLOCK# | NUMBER | Block number |
| STATUS | VARCHAR2(10) | Status of the block:<br>• free - Not currently in use<br>• xcur - Exclusive<br>• scur - Shared current<br>• cr - Consistent read<br>• read - Being read from disk<br>• mrec - In media recovery mode<br>• irec - In instance recovery mode |
| XNC | NUMBER | Number of parallel cache management (PCM) lock conversions due to contention with another instance |
| FORCED_READS | NUMBER | Number of times the block had to be reread from the cache because another instance has forced it out of this instance's cache by requesting the lock on the block in exclusive mode |
| FORCED_WRITES | NUMBER | Number of times GCS had to write this block to cache because this instance had used the block and another instance had requested the lock on the block in a conflicting mode |
| NAME | VARCHAR2(128) | Name of the database object containing the block |
| KIND | VARCHAR2(15) | Type of database object: |

| Column | Datatype | Description |
|---|---|---|
| | | • 1 - Index<br>• 2 - Table<br>• 3 - Cluster<br>• 4 - View<br>• 5 - Synonym<br>• 6 - Sequence<br>• 7 - Procedure<br>• 8 - Function<br>• 9 - Package<br>• 10 - Nonexistent<br>• 11 - Package body<br>• 12 - Trigger<br>• 13 - Type<br>• 14 - Type body<br>• 19 - Table partition<br>• 20 - Index partition<br>• 21 - LOB<br>• 22 - Library<br>• Null - Unknown |
| OWNER# | NUMBER | Owner number |
| LOCK_ELEMENT_ADDR | RAW(4 \| 8) | Address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| LOCK_ELEMENT_NAME | NUMBER | Address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| INDX | NUMBER | Platform-specific lock manager identifier |
| CLASS | NUMBER | Platform-specific lock manager identifier |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CACHE_TRANSFER

V$CACHE_TRANSFER is identical to the V$CACHE view but only displays blocks that have been pinged at least once. This view contains information from the block header of each block in the SGA of the current instance as related to particular database objects. This is an Oracle Real Application Clusters view.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Data file identifier number (to find the file name, query DBA_DATA_FILES or V$DBFILE) |

| Column | Datatype | Description |
|---|---|---|
| BLOCK# | NUMBER | Block number |
| CLASS# | NUMBER | Class number |
| STATUS | VARCHAR2(10) | Status of the block:<br>• free - Not currently in use<br>• xcur - Exclusive<br>• scur - Shared current<br>• cr - Consistent read<br>• read - Being read from disk<br>• mrec - In media recovery mode<br>• irec - In instance recovery mode |
| XNC | NUMBER | Number of PCM lock conversions due to contention with another instance. This column is obsolete and maintained for backward compatibility. |
| FORCED_READS | NUMBER | Number of times the block had to be reread from the cache because another instance has forced it out of this instance's cache by requesting the lock on the block in exclusive mode |
| FORCED_WRITES | NUMBER | Number of times GCS had to write this block to cache because this instance had used the block and another instance had requested the lock on the block in a conflicting mode |
| NAME | VARCHAR2(128) | Name of the database object containing the block |
| PARTITION_NAME | VARCHAR2(128) | NULL for nonpartitioned objects |
| KIND | VARCHAR2(15) | Type of database object<br>**See Also:** Table 8-1 |
| OWNER# | NUMBER | Owner number |
| LOCK_ELEMENT_ADDR | RAW(4 \| 8) | Address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| LOCK_ELEMENT_NAME | NUMBER | The name of the lock that contains the PCM lock that is covering the buffer |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$CACHE"

# V$CIRCUIT

V$CIRCUIT contains information about virtual circuits, which are user connections to the database through dispatchers and servers.

| Column | Datatype | Description |
|---|---|---|
| CIRCUIT | RAW(4 \| 8) | Circuit address |
| DISPATCHER | RAW(4 \| 8) | Current dispatcher process address |
| SERVER | RAW(4 \| 8) | Current server process address |
| WAITER | RAW(4 \| 8) | Address of the server process that is waiting for the (currently busy) circuit to become available |
| SADDR | RAW(4 \| 8) | Address of the session bound to the circuit |
| STATUS | VARCHAR2(16) | Status of the circuit:<br>• BREAK - currently interrupted<br>• EOF - about to be removed<br>• OUTBOUND - an outward link to a remote database<br>• NORMAL - normal circuit into the local database |
| QUEUE | VARCHAR2(16) | Queue the circuit is currently on:<br>• COMMON - on the common queue, waiting to be picked up by a server process<br>• DISPATCHER - waiting for the dispatcher<br>• SERVER - currently being serviced<br>• NONE - idle circuit |
| MESSAGE0 | NUMBER | Size in bytes of the messages in the first message buffer |
| MESSAGE1 | NUMBER | Size in bytes of the messages in the second message buffer |
| MESSAGE2 | NUMBER | Size in bytes of the messages in the third message buffer |
| MESSAGE3 | NUMBER | Size in bytes of the messages in the fourth message buffer |
| MESSAGES | NUMBER | Total number of messages that have gone through this circuit |
| BYTES | NUMBER | Total number of bytes that have gone through this circuit |
| BREAKS | NUMBER | Total number of breaks (interruptions) for this circuit |
| PRESENTATION | VARCHAR2(257) | Presentation protocol used by the client and server |
| PCIRCUIT | RAW(4 \| 8) | Address of the parent circuit |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CLASS_CACHE_TRANSFER

V$CLASS_CACHE_TRANSFER is deprecated. The information that was provided in this view is now provided in the V$INSTANCE_CACHE_TRANSFER and V$SEGMENT_STATISTICS views.

| Column | Datatype | Description |
|---|---|---|
| CLASS | CHAR(10) | Block class; always data block |
| X_2_NULL | NUMBER | Number of blocks with Exclusive-to-NULL conversions; always 0 |
| X_2_NULL_FORCED_WRITE | NUMBER | Number of Exclusive-to-NULL forced writes; always 0 |
| X_2_NULL_FORCED_STALE | NUMBER | Number of Exclusive-to-NULL blocks converted to CR; always 0 |
| X_2_S | NUMBER | Number of blocks with Exclusive-to-Shared conversions; always 0 |
| X_2_S_FORCED_WRITE | NUMBER | Number of Exclusive-to-Shared forced writes; always 0 |
| S_2_NULL | NUMBER | Number of blocks with Shared-to-NULL conversions; always 0 |
| S_2_NULL_FORCED_STALE | NUMBER | Number of Shared-to-NULL blocks converted to CR; always 0 |
| NULL_2_X | NUMBER | Number of blocks with NULL-to-Exclusive conversions; always 0 |
| S_2_X | NUMBER | Number of blocks with Shared-to-Exclusive conversions; always 0 |
| NULL_2_S | NUMBER | Number of blocks with NULL-to-Shared conversions; always 0 |
| CR_TRANSFER | NUMBER | Number of CR blocks transferred; always 0 |
| CURRENT_TRANSFER | NUMBER | Number of current blocks transferred; always 0 |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$INSTANCE_CACHE_TRANSFER"
- "V$SEGMENT_STATISTICS"

# V$CLIENT_SECRETS

V$CLIENT_SECRETS lists the secrets that are present in the keystore. Only SYS, SYSKM, and users with the ADMINISTER KEY MANAGEMENT privilege can access this view.

| Column | Datatype | Description |
|---|---|---|
| CLIENT | VARCHAR2(2000) | Name provided by the client |
| SECRET_TAG | VARCHAR2(4000) | Associated information with the client |
| CREATION_TIME | TIMESTAMP(6) WITH TIME ZONE | Time when the secret was created |
| ACTIVATION_TIME | TIMESTAMP(6) WITH TIME ZONE | Time when the secret was actually put to use |
| OWNER | VARCHAR2(30) | User who created the secret |
| OWNER_ID | NUMBER | User ID of the user who created the secret |
| KEYSTORE_TYPE | VARCHAR2(17) | Secret is in Hardware Security Module (HSM) or Software Key Store |
| BACKED_UP | VARCHAR2(9) | Indicates whether the secret has been backed up or not |
| OWNER_DBNAME | VARCHAR2(30) | Database that created the secret |
| OWNER_DBID | NUMBER | Database ID where the secret was created |
| OWNER_INSTANCE_NAME | VARCHAR2(30) | Instance name of the instance where the secret was created |
| OWNER_INSTANCE_NUMBER | NUMBER | Instance number of the instance where the secret was created |
| OWNER_INSTANCE_SERIAL | NUMBER | Serial number of the instance where the secret was created |
| OWNER_PDBNAME | VARCHAR2(30) | Pluggable database (PDB) where the secret was created |
| OWNER_PDBID | NUMBER | PDB ID where the secret was created |
| OWNER_PDBUID | NUMBER | PDB UID where the secret was created |
| OWNER_PDBGUID | RAW(16) | PDB GUID where the secret was created |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Security Guide* for information about keystore management

# V$CLIENT_STATS

V$CLIENT_STATS displays measures for all sessions that are active for the client identifier per instance. The statistics available in this view are a subset of those available in V$SESSTAT and V$SESS_TIME_MODEL.

| Column | Datatype | Description |
|---|---|---|
| CLIENT_IDENTIFIER | VARCHAR2(64) | Client identifier |
| STAT_ID | NUMBER | Statistic identifier |
| STAT_NAME | VARCHAR2(64) | Derived statistic name from V$STATNAME and V$SESS_TIME_MODEL |
| VALUE | NUMBER | Cumulative value (in microseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SESSTAT"
- "V$SESS_TIME_MODEL"

# V$CLONEDFILE

V$CLONEDFILE provides CloneDB file information.

| Column | Datatype | Description |
|---|---|---|
| SNAPSHOTFILENAME | VARCHAR2(513) | Snapshot/master file name. This file is the master file from the master database. The snapshot file is also sometimes referred to as the srcfile. |
| CLONEFILENAME | VARCHAR2(513) | CloneDB file name. This file is the cloned file (of master file) which resides in the current (cloned database). The cloned file is sometimes referred to as the destfile. |
| SNAPSHOTBLKREAD | NUMBER | Number of blocks reads to the snapshot file (master file) |
| SNAPSHOTREQUEST | NUMBER | Number of read requests to the snapshot file (master file) |
| FILENUMBER | NUMBER | File number of the cloned file |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| BLOCKS_ALLOCATED[1] | NUMBER | Amount of space allocated in blocks in the file system. This is less than or equal to the actual file size. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$CLUSTER_INTERCONNECTS

V$CLUSTER_INTERCONNECTS displays one or more interconnects that are being used for cluster communication.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(15) | Name of the interconnect (such as eth0) |
| IP_ADDRESS | VARCHAR2(16) | IP address of the interconnect |
| IS_PUBLIC | VARCHAR2(3) | If the value is YES, the interface is known to the public.<br>If the value is NO, the interface is known to be private. Note that if the CLUSTER_INTERCONNETS initialization parameter is also specified, then it is expected that the interconnect is private. Oracle expects cluster traffic to be run on private interconnects only.<br>f the value is empty, it is unknown whether the interface is public or private.<br>Oracle recommends that you set the interface for Oracle Real Application Clusters (Oracle RAC) communication in the Oracle Cluster Registry (OCR). |
| SOURCE | VARCHAR2(31) | Indicates where this interface was picked up from:<br>• Oracle Cluster Registry - Interface was configured in the OCR and Oracle Database found the interface in the OCR<br>• Operating-system dependent software - Oracle Database automatically detects this<br>• CLUSTER_INTERCONNECTS parameter - This initialization parameter was set |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CON_SYS_TIME_MODEL

V$CON_SYS_TIME_MODEL displays the systemwide accumulated times for various operations for the container from which it is queried. The time reported is the total elapsed or CPU time (in microseconds). Any timed operation will buffer at most 5 seconds of time data. Specifically, this means that if a timed operation (such as SQL execution) takes a long period of time to perform, the data published to this view is at most missing 5 seconds of the time accumulated for the operation.

The time values are 8-byte integers and can therefore hold approximately 580,000 years worth of time before wrapping. Background process time is not included in a statistic value unless the statistic is specifically for background processes.

| Column | Datatype | Description |
|---|---|---|
| STAT_ID | NUMBER | Statistic identifier for the time statistic |
| STAT_NAME | VARCHAR2(64) | Name of the statistic (see Table 9-1) |
| VALUE | NUMBER | Amount of time (in microseconds) that the system has spent in this operation |
| CON_ID | NUMBER | When queried from a non-CDB, the accumulated times for operations in that instance are returned, and the CON_ID value is set to 0.<br>When queried from the root of a CDB, accumulated times for operations in every container are returned, and the CON_ID value indicates the container to which the times belong.<br>When queried from a PDB, accumulated times for operations in that PDB are returned, and the CON_ID value is set to the container ID for that PDB. |

# V$CON_SYSSTAT

V$CON_SYSSTAT displays system statistics, including OLAP kernel statistics for the container from which it is queried. To find the name of the statistic associated with each statistic number (STATISTIC#), query the V$STATNAME view.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC# | NUMBER | Statistic number<br>**Note:** Statistics numbers are not guaranteed to remain constant from one release to another. Therefore, you should rely on the statistics name rather than its number in your applications. |
| NAME | VARCHAR2(64) | Statistic name. You can get a complete listing of statistic names by querying the V$STATNAME view. |
| CLASS | NUMBER | A number representing one or more statistics class. The following class numbers are additive:<br>• 1 - User<br>• 2 - Redo<br>• 4 - Enqueue<br>• 8 - Cache<br>• 16 - OS<br>• 32 - Real Application Clusters<br>• 64 - SQL |

| Column | Datatype | Description |
|---|---|---|
| | | • 128 - Debug |
| VALUE | NUMBER | Statistic value |
| STAT_ID | NUMBER | Identifier of the statistic |
| CON_ID | NUMBER | When queried from a non-CDB, the statistics in that instance are returned, and the CON_ID value is set to 0.<br>When queried from the root of a CDB, the statistics in every container are returned, and the CON_ID value indicates the container to which the statistics belong.<br>When queried from a PDB, statistics in that PDB are returned, and the CON_ID value is set to the container ID for that PDB. |

**See Also:**

"V$STATNAME" and Statistics Descriptions

# V$CON_SYSTEM_EVENT

V$CON_SYSTEM_EVENT displays information on total waits for an event in a container. Note that the TIME_WAITED and AVERAGE_WAIT columns will contain a value of zero on those platforms that do not support a fast timing mechanism. If you are running on one of these platforms and you want this column to reflect true wait times, then you must set TIMED_STATISTICS to TRUE in the parameter file; doing this will have a small negative effect on system performance.

**See Also:**

"TIMED_STATISTICS"

| Column | Datatype | Description |
|---|---|---|
| EVENT | VARCHAR2(64) | Name of the wait event |
| TOTAL_WAITS | NUMBER | Total number of waits for the event |
| TOTAL_TIMEOUTS | NUMBER | Total number of timeouts for the event |
| TIME_WAITED | NUMBER | Total amount of time waited for the event (in hundredths of a second) |
| AVERAGE_WAIT | NUMBER | Average amount of time waited for the event (in hundredths of a second) |
| TIME_WAITED_MICRO | NUMBER | Total amount of time waited for the event (in microseconds) |
| TOTAL_WAITS_FG | NUMBER | Total number of waits for the event, from foreground sessions |
| TOTAL_TIMEOUTS_FG | NUMBER | Total number of timeouts for the event, from foreground sessions |
| TIME_WAITED_FG | NUMBER | Amount of time waited for the event (in hundredths of a second), from foreground sessions |

| Column | Datatype | Description |
|---|---|---|
| AVERAGE_WAIT_FG | NUMBER | Average amount of time waited for the event (in hundredths of a second), from foreground sessions |
| TIME_WAITED_MICRO_FG | NUMBER | Amount of time waited for the event (in microseconds), from foreground sessions |
| EVENT_ID | NUMBER | Identifier of the wait event |
| WAIT_CLASS_ID | NUMBER | Identifier of the class of the wait event |
| WAIT_CLASS# | NUMBER | Number of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | Name of the class of the wait event |
| CON_ID | NUMBER | When queried from a non-CDB, the wait event data in that instance are returned, and the CON_ID value is set to 0.<br>When queried from the root of a CDB, the wait event data in every container is returned, and the CON_ID value indicates the container to which the wait event data belong.<br>When queried from a PDB, wait event data in that PDB are returned, and the CON_ID value is set to the container ID for that PDB. |

# V$CON_SYSTEM_WAIT_CLASS

V$CON_SYSTEM_WAIT_CLASS displays the time totals for each registered wait class in a container.

| Column | Datatype | Description |
|---|---|---|
| WAIT_CLASS_ID | NUMBER | Identifier of the wait class |
| WAIT_CLASS# | NUMBER | Number of the wait class |
| WAIT_CLASS | VARCHAR2(64) | Name of the wait class |
| TOTAL_WAITS | NUMBER | Number of times waits of the class occurred |
| TIME_WAITED | NUMBER | Amount of time (in hundredths of a second) spent in the wait by all sessions |
| TOTAL_WAITS_FG | NUMBER | Number of times waits from this wait class occurred in foreground sessions |
| TIME_WAITED_FG | NUMBER | Amount of time (in hundredths of a second) spent in waits from this wait class in foreground sessions |
| CON_ID | NUMBER | When queried from a non-CDB, time totals for each registered wait class in that instance are returned, and the CON_ID value is set to 0.<br>When queried from the root of a CDB, time totals for each registered wait class in every container is returned, and the CON_ID value indicates the container to which the time total belong.<br>When queried from a PDB, time totals for each registered wait class in that PDB are returned, and the CON_ID value is set to the container ID for that PDB. |

# V$CONFIGURED_INTERCONNECTS

V$CONFIGURED_INTERCONNECTS displays all the interconnects that Oracle is aware of. This view attempts to answer the question of where Oracle found the information about a specific interconnect.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(15) | Name of the interconnect (such as eth0) |
| IP_ADDRESS | VARCHAR2(16) | IP address of the interconnect |
| IS_PUBLIC | VARCHAR2(3) | If the value is YES, the interface is known to the public.<br>If the value is NO, the interface is known to be private. Note that if the CLUSTER_INTERCONNECTS initialization parameter is also specified, then it is expected that the interconnect is private. Oracle expects cluster traffic to be run on private interconnects only.<br>If the value is empty, it is unknown whether the interface is public or private.<br>Oracle recommends that you set the interface for Oracle Real Application Clusters (Oracle RAC) communication in the Oracle Cluster Registry (OCR). |
| SOURCE | VARCHAR2(31) | Indicates where this interface was picked up from:<br>• Oracle Cluster Registry - Interface was configured in the OCR and Oracle Database found the interface in the OCR<br>• Operating-system dependent software - Oracle Database automatically detects this<br>• CLUSTER_INTERCONNECTS parameter - This initialization parameter was set |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CONTAINERS

V$CONTAINERS displays information about PDBs and the root associated with the current instance.

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

| Column | Datatype | Description |
|---|---|---|
| DBID | NUMBER | PDB identifier calculated when the PDB is created and stored in all file headers associated with the PDB |
| CON_UID | NUMBER | Unique identifier associated with the PDB |
| GUID | RAW(16) | Globally unique identifier (GUID) of this PDB |
| NAME | VARCHAR2(30) | Name of the PDB |
| OPEN_MODE | VARCHAR2(10) | Open mode information. Possible values:<br>• MOUNTED<br>• READ WRITE<br>• READ ONLY<br>• MIGRATE |
| RESTRICTED | VARCHAR2(3) | Indicates whether only users possessing RESTRICTED SESSION privilege can connect to the PDB |
| OPEN_TIME | TIMESTAMP(3) | Date and time when the database was last opened |
| CREATE_SCN | NUMBER | System change number (SCN) for the creation of this PDB |
| TOTAL_SIZE | NUMBER | If a PDB is opened, disk space (in bytes) used by the container, including both data and temp files. If a PDB is closed, will be set to 0. |
| BLOCK_SIZE | NUMBER | The current block size for the PDB |
| RECOVERY_STATUS[1] | VARCHAR2(8) | Shows whether recovery is enabled or disabled for the PDB. Possible values:<br>• ENABLED<br>• DISABLED |
| SNAPSHOT_PARENT_CON_ID[1] | NUMBER | This column shows the container ID of the master PDB that this PDB is a snapshot clone of. This column shows a non-zero value only if the PDB is a snapshot clone. For all other cases, it shows a value of 0. |

1 This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$CONTEXT

V$CONTEXT displays set attributes in the current session.

| Column | Datatype | Description |
|---|---|---|
| NAMESPACE | VARCHAR2(31) | Namespace that the attribute is in |
| ATTRIBUTE | VARCHAR2(31) | Name of the attribute |
| VALUE | VARCHAR2(4000) | Value of the attribute |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CONTROLFILE

V$CONTROLFILE displays the names of the control files.

| Column | Datatype | Description |
|---|---|---|
| STATUS | VARCHAR2(7) | INVALID if the name cannot be determined (which should not occur); NULL if the name can be determined |
| NAME | VARCHAR2(513) | Name of the control file |
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether the file was created in the fast recovery area (YES) or not (NO) |
| BLOCK_SIZE | NUMBER | Control file block size |
| FILE_SIZE_BLKS | NUMBER | Control file size (in blocks) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CONTROLFILE_RECORD_SECTION

V$CONTROLFILE_RECORD_SECTION displays information about the control file record sections.

| Column | Datatype | Description |
|---|---|---|
| TYPE | VARCHAR2(28) | Identifies the type of record section:<br>• DATABASE<br>• CKPT PROGRESS<br>• REDO THREAD<br>• REDO LOG<br>• DATAFILE<br>• FILENAME<br>• TABLESPACE<br>• TEMPORARY FILENAME<br>• RMAN CONFIGURATION<br>• LOG HISTORY<br>• OFFLINE RANGE<br>• ARCHIVED LOG<br>• BACKUP SET<br>• BACKUP PIECE<br>• BACKUP DATAFILE |

| Column | Datatype | Description |
|---|---|---|
| | | • BACKUP REDOLOG<br>• DATAFILE COPY<br>• BACKUP CORRUPTION<br>• COPY CORRUPTION<br>• DELETED OBJECT<br>• PROXY COPY<br>• BACKUP SPFILE<br>• DATABASE INCARNATION<br>• FLASHBACK LOG<br>• RECOVERY DESTINATION<br>• INSTANCE SPACE RESERVATION<br>• REMOVABLE RECOVERY FILES<br>• RMAN STATUS<br>• THREAD INSTANCE NAME MAPPING<br>• MTTR<br>• DATAFILE HISTORY<br>• PLUGGED IN DATAFILE |
| RECORD_SIZE | NUMBER | Record size in bytes |
| RECORDS_TOTAL | NUMBER | Number of records allocated for the section |
| RECORDS_USED | NUMBER | Number of records used in the section |
| FIRST_INDEX | NUMBER | Index (position) of the first record |
| LAST_INDEX | NUMBER | Index of the last record |
| LAST_RECID | NUMBER | Record ID of the last record |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$COPY_CORRUPTION

V$COPY_CORRUPTION displays information about data file copy corruptions from the control file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Copy corruption record ID |
| STAMP | NUMBER | Copy corruption record stamp |
| COPY_RECID | NUMBER | Data file copy record ID |
| COPY_STAMP | NUMBER | Data file copy record stamp |
| FILE# | NUMBER | Data file number |

| Column | Datatype | Description |
|---|---|---|
| BLOCK# | NUMBER | First block of the corrupted range |
| BLOCKS | NUMBER | Number of contiguous blocks in the corrupted range |
| CORRUPTION_CHANGE# | NUMBER | Change number at which the logical corruption was detected. Set to 0 to indicate media corruption. |
| MARKED_CORRUPT | VARCHAR2(3) | (YES \| NO) If set to YES the blocks were not marked corrupted in the data file, but were detected and marked as corrupted while making the data file copy |
| CORRUPTION_TYPE | VARCHAR2(9) | Type of block corruption in the data file:<br>• ALL ZERO - Block header on disk contained only zeros. The block may be valid if it was never filled and if it is in an Oracle7 file. The buffer will be reformatted to the Oracle8 standard for an empty block.<br>• FRACTURED - Block header looks reasonable, but the front and back of the block are different versions.<br>• CHECKSUM - optional check value shows that the block is not self-consistent. It is impossible to determine exactly why the check value fails, but it probably fails because sectors in the middle of the block are from different versions.<br>• CORRUPT - Block is wrongly identified or is not a data block (for example, the data block address is missing)<br>• LOGICAL - Block is logically corrupt<br>• NOLOGGING - Block does not have redo log entries (for example, NOLOGGING operations on primary database can introduce this type of corruption on a physical standby) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$COPY_NONLOGGED

V$COPY_NONLOGGED displays information about nonlogged block ranges in data file copy blocks recorded in the control file.

| Column | Datatype | Description |
|---|---|---|
| INST_ID | NUMBER | Instance ID |
| RECID | NUMBER | Nonlogged copy record ID |
| STAMP | NUMBER | Nonlogged copy record stamp |
| COPY_RECID | NUMBER | Data file copy record ID |
| COPY_STAMP | NUMBER | Data file copy record stamp |
| FILE# | NUMBER | Absolute file number of the data file that contains this range of nonlogged blocks |

| Column | Datatype | Description |
|---|---|---|
| BLOCK# | NUMBER | Block number of the first nonlogged block in the range of nologged blocks |
| BLOCKS | NUMBER | Number of nonlogged blocks found starting with BLOCK# |
| NONLOGGED_CHANGE# | NUMBER | The smallest SCN on which any block in this block range became nonlogged. NULL if unknown. |
| NONLOGGED_TIME | VARCHAR2 | The time that corresponds to NONLOGGED_CHANGE#. NULL if unknown. |
| RESETLOGS_CHANGE# | VARCHAR2 | The resetlogs SCN of the incarnation on which this block range was first marked as nonlogged. NULL if unknown. |
| RESETLOGS_TIME | VARCHAR2 | The resetlogs time of the incarnation on which this block range was first marked as nologged. NULL if unknown. |
| OBJECT# | VARCHAR2 | The object ID this range belongs to. If this field is NULL, the object number is unknown. |
| REASON | CHAR(7) | The reason why this block range appears in this list. For Oracle Database 12*c*, it is always UNKNOWN. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CORRUPT_XID_LIST

V$CORRUPT_XID_LIST displays all corrupted XIDs.

| Column | Datatype | Description |
|---|---|---|
| CORRUPT_XID | VARCHAR2(256) | Name of corrupt XID |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CPOOL_CC_INFO

V$CPOOL_CC_INFO displays information about the pool-to-connection class mapping for the Database Resident Connection Pool per instance.

| Column | Datatype | Description |
|---|---|---|
| POOL_NAME | VARCHAR2(1024) | Name of the Database Resident Connection Pool |
| CCLASS_NAME | VARCHAR2(1024) | Name of the connection class |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CPOOL_CC_STATS

V$CPOOL_CC_STATS displays information about the connection class level statistics for the Database Resident Connection Pool per instance.

| Column | Datatype | Description |
|---|---|---|
| CCLASS_NAME | VARCHAR2(1024) | Name of the connection class |
| NUM_REQUESTS | NUMBER | Number of session requests |
| NUM_HITS | NUMBER | Total number of times a session that matches with the request was found in the pool |
| NUM_MISSES | NUMBER | Total number of times an exact match to the request was not found in the pool and a new session had to be created |
| NUM_WAITS | NUMBER | Total number of times session requests had to wait before getting served |
| WAIT_TIME | NUMBER | Reserved for future use |
| CLIENT_REQ_TIMEOUTS | NUMBER | Reserved for future use |
| NUM_AUTHENTICATIONS | NUMBER | Total number of authentications of clients done by the pool |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CPOOL_CONN_INFO

V$CPOOL_CONN_INFO displays connection information about each connection to the connection broker.

| Column | Datatype | Description |
|---|---|---|
| CMON_ADDR | RAW(4 \| 8) | Address of the connection broker |
| SESSION_ADDR | RAW(4 \| 8) | Address of the session associated with the connection; NULL if the connection does not have an associated session |
| CONNECTION_ADDR | RAW(4 \| 8) | Address of the connection |
| USERNAME | VARCHAR2(1024) | Name of the user associated with the connection |
| PROXY_USER | VARCHAR2(1024) | Name of the proxy user |
| CCLASS_NAME | VARCHAR2(1024) | Connection class associated with the connection |
| PURITY | VARCHAR2(1024) | Purity used to create the connection (can be SELF or NEW) |
| TAG | VARCHAR2(1024) | Tag, if specified, at connection creation time |
| SERVICE | VARCHAR2(64) | TNS service name for the connection |
| PROCESS_ID | VARCHAR2(24) | Client process ID of the process which created the connection |
| PROGRAM | VARCHAR2(48) | Program name of the client process which created the connection |
| MACHINE | VARCHAR2(64) | Machine name of the client process which created the connection |
| TERMINAL | VARCHAR2(30) | Terminal identifier of the client process which created the connection |
| CONNECTION_MODE | VARCHAR2(1024) | Reserved for internal use |
| CONNECTION_STATUS | VARCHAR2(10) | Status of the connection:<br>• NONE<br>• CONNECTING<br>• ACTIVE<br>• WAITING<br>• IDLE<br>• CLOSING |
| CLIENT_REGID | NUMBER | Query cache registration ID sent by the client |
| CURSTATUS_TIME | NUMBER | Time in microseconds spent in the current state. See CONNECTION_STATUS above. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CPOOL_STATS

V$CPOOL_STATS displays information about the Database Resident Connection Pool statistics for an instance.

| Column | Datatype | Description |
|---|---|---|
| POOL_NAME | VARCHAR2(1024) | Name of the Database Resident Connection Pool |
| NUM_OPEN_SERVERS | NUMBER | Total number of busy and free servers in the pool (including the authentication servers) |
| NUM_BUSY_SERVERS | NUMBER | Total number of busy servers in the pool (not including the authentication servers) |
| NUM_AUTH_SERVERS | NUMBER | Number of authentication servers in the pool |
| NUM_REQUESTS | NUMBER | Number of client requests |
| NUM_HITS | NUMBER | Total number of times client requests found matching pooled servers in the pool |
| NUM_MISSES | NUMBER | Total number of times client requests could not find a matching pooled server in the pool |
| NUM_WAITS | NUMBER | Total number of client requests that had to wait due to non-availability of free pooled servers |
| WAIT_TIME | NUMBER | Reserved for future use |
| CLIENT_REQ_TIMEOUTS | NUMBER | Reserved for future use |
| NUM_AUTHENTICATIONS | NUMBER | Total number of authentications of clients done by the pool |
| NUM_PURGED | NUMBER | Total number of sessions purged by the pool |
| HISTORIC_MAX | NUMBER | Maximum size that the pool has ever reached |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$CR_BLOCK_SERVER

V$CR_BLOCK_SERVER displays statistics on the Global Cache Service processes (LMS) used in cache fusion.

| Column | Datatype | Description |
|---|---|---|
| CR_REQUESTS | NUMBER | Number of CR blocks served due to remote CR block requests |

| Column | Datatype | Description |
|---|---|---|
| CURRENT_REQUESTS | NUMBER | Number of current blocks served due to remote CR block requests<br>CR_REQUESTS + CURRENT_REQUESTS = global cache CR clocks served (from V$SYSSTAT). |
| DATA_REQUESTS | NUMBER | Number of current or CR requests for data blocks |
| UNDO_REQUESTS | NUMBER | Number of CR requests for undo blocks |
| TX_REQUESTS | NUMBER | Number of CR requests for undo segment header blocks |
| OTHER_REQUESTS | NUMBER | Number of CR requests for other types of blocks<br>DATA_REQUESTS + UNDO_REQUESTS + TX_REQUESTS + OTHER_REQUESTS = total number of requests handled by the LMS processes |
| CURRENT_RESULTS | NUMBER | Number of requests for which no changes were rolled out of the block returned to the requesting instance |
| PRIVATE_RESULTS | NUMBER | Number of requests for which changes were rolled out of the block returned to the requesting instance, and only the requesting transaction can use the resulting CR block |
| ZERO_RESULTS | NUMBER | Number of requests for which changes were rolled out of the block returned to the requesting instance. Only zero-XID transactions can use the block. |
| DISK_READ_RESULTS | NUMBER | Number of requests for which the requesting instance had to read the requested block from disk |
| FAIL_RESULTS | NUMBER | Number of requests that failed; the requesting transaction must reissue the request |
| STALE | NUMBER | Number of requests for which the disk read of the requested block was stale |
| FAIRNESS_DOWN_CONVERTS | NUMBER | Number of times an instance receiving a request has down-converted an X lock on a block because it was not modifying the block |
| FAIRNESS_CLEARS | NUMBER | Number of times the "fairness counter" was cleared. This counter tracks the number of times a block was modified after it was served. |
| FREE_GC_ELEMENTS | NUMBER | Number of times a request was received from another instance and the X lock had no buffers |
| FLUSHES | NUMBER | Number of times the log has been flushed by an LMS process |
| FLUSHES_QUEUED | NUMBER | Number of flushes queued by an LMS process |
| FLUSH_QUEUE_FULL | NUMBER | Number of times the flush queue was full |
| FLUSH_MAX_TIME | NUMBER | Maximum time for flush |
| LIGHT_WORKS | NUMBER | Number of times the light-work rule was evoked. This rule prevents the LMS processes from going to disk while responding to CR requests for data, undo, or undo segment header blocks. |

| Column | Datatype | Description |
|---|---|---|
| | | This rule can prevent the LMS process from completing its response to the CR request. |
| ERRORS | NUMBER | Number of times an error was signalled by an LMS process |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view contains internal diagnostic information for use by Oracle Support Services. It is subject to change without notice.

# V$CURRENT_BLOCK_SERVER

V$CURRENT_BLOCK_SERVER displays statistics on the Global Cache Service processes (lMS) used in cache fusion.

| Column | Datatype | Description |
|---|---|---|
| PIN1 | NUMBER | Pins taking less than 1 millisecond |
| PIN10 | NUMBER | Pins taking 1 to 10 milliseconds |
| PIN100 | NUMBER | Pins taking 10 to 100 milliseconds |
| PIN1000 | NUMBER | Pins taking 100 to 1000 milliseconds |
| PIN10000 | NUMBER | Pins taking 1000 to 10000 milliseconds |
| FLUSH1 | NUMBER | Flushes taking less than 1 millisecond |
| FLUSH10 | NUMBER | Flushes taking 1 to 10 milliseconds |
| FLUSH100 | NUMBER | Flushes taking 10 to 100 milliseconds |
| FLUSH1000 | NUMBER | Flushes taking 100 to 1000 milliseconds |
| FLUSH10000 | NUMBER | Flushes taking 1000 to 10000 milliseconds |
| WRITE1 | NUMBER | Writes taking less than 1 millisecond |
| WRITE10 | NUMBER | Writes taking 1 to 10 milliseconds |
| WRITE100 | NUMBER | Writes taking 10 to 100 milliseconds |
| WRITE1000 | NUMBER | Writes taking 100 to 1000 milliseconds |
| WRITE10000 | NUMBER | Writes taking 1000 to 10000 milliseconds |
| CLEANDC | NUMBER | Reserved for internal use |

| Column | Datatype | Description |
|---|---|---|
| RCVDC | NUMBER | Number of lock down-converts to S (shared) caused by instance recovery |
| QUEUEDC | NUMBER | Number of queued lock down-converts to NULL |
| EVICTDC | NUMBER | Number of lock down-converts to NULL caused by an SGA shrink |
| WRITEDC | NUMBER | Number of dirty blocks in read-mostly objects which were written and the X (exclusive) lock down-converted to S (shared) locks |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DATABASE

V$DATABASE displays information about the database from the control file.

| Column | Datatype | Description |
|---|---|---|
| DBID | NUMBER | Database identifier calculated when the database is created and stored in all file headers |
| NAME | VARCHAR2(9) | Name of the database |
| CREATED | DATE | Creation date of the database. If the control file was re-created using the CREATE CONTROLFILE statement, then this column displays the date that the control file was re-created. |
| RESETLOGS_CHANGE# | NUMBER | System change number (SCN) at open resetlogs |
| RESETLOGS_TIME | DATE | Timestamp of open resetlogs |
| PRIOR_RESETLOGS_CHANGE# | NUMBER | SCN at prior resetlogs |
| PRIOR_RESETLOGS_TIME | DATE | Timestamp of prior resetlogs |
| LOG_MODE | VARCHAR2(12) | Archive log mode:<br>• NOARCHIVELOG<br>• ARCHIVELOG<br>• MANUAL |
| CHECKPOINT_CHANGE# | NUMBER | Last SCN checkpointed |
| ARCHIVE_CHANGE# | NUMBER | Database force archiving SCN. Any redo log with a start SCN below this will be forced to archive out. |
| CONTROLFILE_TYPE | VARCHAR2(7) | Type of control file:<br>• STANDBY - Indicates that the database is in standby mode |

| Column | Datatype | Description |
|---|---|---|
| | | • CLONE - Indicates a clone database<br>• BACKUP \| CREATED - Indicates the database is being recovered using a backup or created control file<br>• CURRENT - database is available for general use |
| CONTROLFILE_CREATED | DATE | Creation date of the control file |
| CONTROLFILE_SEQUENCE# | NUMBER | Control file sequence number incremented by control file transactions |
| CONTROLFILE_CHANGE# | NUMBER | Last SCN in backup control file; null if the control file is not a backup |
| CONTROLFILE_TIME | DATE | Last timestamp in backup control file; null if the control file is not a backup |
| OPEN_RESETLOGS | VARCHAR2(11) | (NOT ALLOWED \| ALLOWED \| REQUIRED) Indicates whether the next database open allows or requires the resetlogs option |
| VERSION_TIME | DATE | Version time |
| OPEN_MODE | VARCHAR2(20) | Open mode information:<br>• MOUNTED<br>• READ WRITE<br>• READ ONLY<br>• READ ONLY WITH APPLY - A physical standby database is open in real-time query mode |
| PROTECTION_MODE | VARCHAR2(20) | Protection mode currently in effect for the database:<br>• MAXIMUM PROTECTION - Database is running in maximized protection mode<br>• MAXIMUM AVAILABILITY - Database is running in maximized availability mode<br>• RESYNCHRONIZATION - Database is running in resynchronization mode<br>• MAXIMUM PERFORMANCE - Database is running in maximized performance mode<br>• UNPROTECTED - Database is unprotected (this normally occurs when the primary database is mounted and not open) |
| PROTECTION_LEVEL | VARCHAR2(20) | Aggregated protection mode currently in effect for the database:<br>• MAXIMUM PROTECTION - Database is running in maximized protection mode<br>• MAXIMUM AVAILABILITY - Database is running in maximized availability mode<br>• RESYNCHRONIZATION - Database is running in resynchronization mode<br>• MAXIMUM PERFORMANCE - Database is running in maximized performance mode<br>• UNPROTECTED - Database is unprotected (this normally occurs when the primary database is mounted and not open)<br>**Note:** This column is an aggregation of the PROTECTION_MODE of all standby archive log destinations. |
| REMOTE_ARCHIVE | VARCHAR2(8) | Value of the REMOTE_ARCHIVE_ENABLE initialization parameter |

| Column | Datatype | Description |
|---|---|---|
| ACTIVATION# | NUMBER | Number assigned to the database instantiation |
| SWITCHOVER# | NUMBER | Number assigned to the database switchover |
| DATABASE_ROLE | VARCHAR2(16) | Current role of the database:<br>• SNAPSHOT STANDBY<br>• LOGICAL STANDBY<br>• PHYSICAL STANDBY<br>• PRIMARY<br>• FAR SYNC |
| ARCHIVELOG_CHANGE# | NUMBER | Highest NEXT_CHANGE# (from the V$ARCHIVED_LOG view) for an archive log |
| ARCHIVELOG_COMPRESSION | VARCHAR2(8) | Status of the archive log compression (ENABLED) or (DISABLED) |
| SWITCHOVER_STATUS | VARCHAR2(20) | Indicates whether switchover is allowed:<br>• NOT ALLOWED - On a primary database, this status indicates that there are no valid and enabled standby databases. On a standby database, this status indicates that a switchover request has not been received from the primary database.<br>• SESSIONS ACTIVE - The database has active sessions. On a physical standby database, the WITH SESSION SHUTDOWN SQL clause must be specified to perform a role transition while in this state. On a logical standby database, a role transition can be performed while in this state, but the role transition will not complete until all current transactions have committed.<br>• SWITCHOVER PENDING - On a physical standby database, this status indicates that a switchover request has been received from the primary database and is being processed. A physical standby database cannot switch to the primary role while in this transient state.<br>• SWITCHOVER LATENT - On a physical standby database, this status indicates that a switchover request was pending, but the original primary database has been switched back to the primary role.<br>• TO PRIMARY - The database is ready to switch to the primary role.<br>• TO STANDBY - The database is ready to switch to either the physical or logical standby role.<br>• TO LOGICAL STANDBY - The database has received a data dictionary from a logical standby database and is ready to switch to the logical standby role.<br>• RECOVERY NEEDED - On a physical standby database, this status indicates that additional redo must be applied before the database can switch to the primary role.<br>• PREPARING SWITCHOVER - On a primary database, this status indicates that a data dictionary is being received from a logical standby database in preparation for switching to the logical standby role. On a logical standby database, this status indicates that the data dictionary has been sent to the primary database and other standby databases.<br>• PREPARING DICTIONARY - On a logical standby database, this status indicates that the data dictionary is being sent to |

| Column | Datatype | Description |
|---|---|---|
| | | the primary database and other standby databases in preparation for switching to the primary role.<br>• FAILED DESTINATION - On a primary database, this status indicates that one or more standby destinations are in an error state.<br>• RESOLVABLE GAP - On a primary database, this status indicates that one or more standby databases have a redo gap that can be automatically resolved by fetching the missing redo from the primary database or from another standby database.<br>• UNRESOLVABLE GAP - On a primary database, this status indicates that one or more standby databases have a redo gap that cannot be automatically resolved by fetching the missing redo from the primary database or from another standby database.<br>• LOG SWITCH GAP - On a primary database, this status indicates that one or more standby databases are missing redo due to a recent log switch. |
| DATAGUARD_BROKER | VARCHAR2(8) | Data Guard broker information:<br>• ENABLED - Database is part of a broker configuration and broker management of the database is enabled<br>• DISABLED - Database is part of a broker configuration and broker management of the database is disabled. This value is displayed if the user disabled broker management of the database or configuration, or if broker management was disabled due to a role change (for example, the old primary was disabled after a failover operation). |
| GUARD_STATUS | VARCHAR2(7) | Protects data from being changed:<br>• ALL - Indicates all users other than SYS are prevented from making changes to any data in the database.<br>• STANDBY - Indicates all users other than SYS are prevented from making changes to any database object being maintained by logical standby.<br>• NONE - Indicates normal security for all data in the database. |
| SUPPLEMENTAL_LOG_DATA_MIN | VARCHAR2(8) | Ensures that LogMiner (and any products building on LogMiner technology) will have sufficient information to support chained rows and various storage arrangements such as cluster tables:<br>• NO - None of the database-wide supplemental logging directives are enabled.<br>In a CDB, a value of NO means that minimal supplemental logging is not enabled in all of the PDBs in the CDB.<br>• IMPLICIT - Minimal supplemental logging is enabled because all or a combination of primary key, unique key, and foreign key supplemental logging is enabled<br>• YES - Minimal supplemental logging is enabled through an ALTER DATABASE ADD SUPPLEMENTAL LOG DATA statement.<br>In a CDB, a value of YES means that minimal supplemental logging is enabled in all of the PDBs in the CDB.<br>**See Also:** *Oracle Database SQL Language Reference* for additional information about the ALTER DATABASE ADD SUPPLEMENTAL LOG DATA statement |

| Column | Datatype | Description |
|---|---|---|
| SUPPLEMENTAL_LOG_DATA_PK | VARCHAR2(3) | For all tables with a primary key, indicates whether all columns of the primary key are placed into the redo log whenever an update is performed (YES) or not (NO).<br>When a value of YES appears in a CDB, it means that primary key supplemental logging is enabled in all of the PDBs in the CDB.<br>When a value of NO appears in a CDB, query the PRIMARY_KEY column in the DBA_SUPPLEMENTAL_LOGGING view for each PDB in the CDB to see whether primary key supplemental logging is enabled in the PDB.<br>**See Also:** *Oracle Database SQL Language Reference* for more information about the ALTER DATABASE ADD SUPPLEMENTAL LOG *supplemental_id_key_clause* statement |
| SUPPLEMENTAL_LOG_DATA_UI | VARCHAR2(3) | For all tables with a unique key, indicates whether all other columns belonging to the unique key are placed into the redo log if any of the unique key columns are modified (YES) or not (NO).<br>When a value of YES appears in a CDB, it means that this value is enabled in all of the PDBs in the CDB.<br>When a value of NO appears in a CDB, query the UNIQUE_INDEX column in the DBA_SUPPLEMENTAL_LOGGING view for each PDB in the CDB to see whether unique column supplemental logging is enabled in the PDB.<br>**See Also:** *Oracle Database SQL Language Reference* for more information about the ALTER DATABASE ADD SUPPLEMENTAL LOG *supplemental_id_key_clause* statement |
| FORCE_LOGGING | VARCHAR2(39) | Indicates whether the database is under force logging mode (YES) or not (NO) |
| PLATFORM_ID | NUMBER | Platform identification number of the database |
| PLATFORM_NAME | VARCHAR2(101) | Platform name of the database |
| RECOVERY_TARGET_INCARNATION# | NUMBER | Incarnation number where all data files are recovered by the RECOVER DATABASE command |
| LAST_OPEN_INCARNATION# | NUMBER | Record number of the incarnation in V$DATABASE_INCARNATION that was last opened successfully |
| CURRENT_SCN | NUMBER | Current SCN; null if the database is not currently open. For a standby database, it is the checkpoint SCN of the mounted physical standby database during media recovery and is always less than the last applied SCN tracked in V$RECOVERY_PROGRESS. |
| FLASHBACK_ON | VARCHAR2(18) | Possible values are as follows:<br>• YES - Flashback is on<br>• NO - Flashback is off<br>• RESTORE POINT ONLY - Flashback is on but one can only flashback to guaranteed restore points |
| SUPPLEMENTAL_LOG_DATA_FK | VARCHAR2(3) | For all tables with a foreign key, indicates whether all other columns belonging to the foreign key are placed into the redo log if any foreign key columns are modified (YES) or not (NO).<br>When a value of YES appears in a CDB, it means that foreign key supplemental logging is enabled in all of the PDBs in the CDB. |

| Column | Datatype | Description |
|---|---|---|
| | | When a value of NO appears in a CDB, query the FOREIGN_KEY column in the DBA_SUPPLEMENTAL_LOGGING view for each PDB in the CDB to see whether foreign key supplemental logging is enabled in the PDB.<br>**See Also:** *Oracle Database SQL Language Reference* for more information about the ALTER DATABASE ADD SUPPLEMENTAL LOG *supplemental_id_key_clause* statement |
| SUPPLEMENTAL_LOG_DATA_ALL | VARCHAR2(3) | For all columns, indicates whether all the fixed-length maximum size columns of that row are placed into the redo log (YES) or not (NO).<br>When a value of YES appears in a CDB, it means that all column supplemental logging is enabled in all of the PDBs in the CDB.<br>When a value of NO appears in a CDB, query the ALL_COLUMN column in the DBA_SUPPLEMENTAL_LOGGING view for each PDB in the CDB to see whether all column supplemental logging is enabled in the PDB.<br>**See Also:** *Oracle Database SQL Language Reference* for more information about the ALTER DATABASE ADD SUPPLEMENTAL LOG *supplemental_id_key_clause* statement |
| DB_UNIQUE_NAME | VARCHAR2(30) | Unique database name |
| STANDBY_BECAME_PRIMARY_SCN | NUMBER | SCN at which a physical standby database became a primary database. This SCN is useful for converting a failed primary database into a physical standby database after a forced failover.<br>**See Also:** *Oracle Data Guard Concepts and Administration* for more information about Oracle Data Guard. |
| FS_FAILOVER_STATUS | VARCHAR2(22) | Fast-start failover status:<br>• DISABLED<br>• BYSTANDER<br>• SYNCHRONIZED<br>• UNSYNCHRONIZED<br>• SUSPENDED<br>• STALLED<br>• LOADING DICTIONARY<br>• PRIMARY UNOBSERVED<br>• REINSTATE REQUIRED<br>• REINSTATE FAILED<br>• TARGET OVER LAG LIMIT<br>• TARGET UNDER LAG LIMIT<br>**See Also:** *Oracle Data Guard Broker* for detailed descriptions of these values<br>**Note:** If the value of this column is DISABLED, then the values for the FS_FAILOVER_CURRENT_TARGET, FS_FAILOVER_THRESHOLD, FS_FAILOVER_OBSERVER_PRESENT, and FS_FAILOVER_OBSERVER_HOST columns in this table are not meaningful. |
| FS_FAILOVER_CURRENT_TARGET | VARCHAR2(30) | DB_UNIQUE_NAME of the standby that is the current FSFO target standby for the Data Guard configuration |

| Column | Datatype | Description |
|---|---|---|
| FS_FAILOVER_THRESHOLD | NUMBER | Time (in seconds) that the observer will attempt to reconnect with a disconnected primary before attempting FSFO with the target standby |
| FS_FAILOVER_OBSERVER_PRESENT | VARCHAR2(7) | Indicates whether the observer is currently connected to the local database (YES) or not (NO)<br>**Note:** This column is consistent throughout an Oracle RAC environment; that is, if the observer is connected to any instance, then all instances will show a value of YES. |
| FS_FAILOVER_OBSERVER_HOST | VARCHAR2(512) | Machine name that is currently hosting the observer process |
| CONTROLFILE_CONVERTED | VARCHAR2(3) | Indicates whether the control file was implicitly converted from its original type during restore (YES) or not (NO)<br>This column will be set to YES when RMAN restores a standby control file from a backup of the control file taken at the primary database or restores a backup control file from a backup taken at the physical standby database.<br>This column will change to NO when the file names are fixed using information in the recovery catalog schema. |
| PRIMARY_DB_UNIQUE_NAME | VARCHAR2(30) | For any Standby database (Physical, Logical, or Snapshot), this column will contain the DB_UNIQUE_NAME of the Primary database that this Standby last received current redo from.<br>If this standby has not received any current redo since last being started, then this column will be null.<br>For a Primary database that had previously been a Standby, this column will contain the DB_UNIQUE_NAME of the last Primary that this database received current redo from while acting as a Standby.<br>For a Primary database that has never been a Standby, this column will be null. |
| SUPPLEMENTAL_LOG_DATA_PL | VARCHAR2(3) | Indicates whether additional information is logged in the redo log (YES) or not (NO) during invocation of procedures in Oracle-supplied packages for which procedural replication is supported.<br>When a value of YES appears in a CDB, it means that supplemental logging for procedural replication is enabled in all of the PDBs in the CDB.<br>When a value of NO appears in a CDB, query the PROCEDURAL column in the DBA_SUPPLEMENTAL_LOGGING view for each PDB in the CDB to see whether supplemental logging for procedural replication is supported in the PDB.<br>**See Also:** *Oracle Data Guard Concepts and Administration* for a list of Oracle-supplied packages that are procedurally replicated to a logical standby database |
| MIN_REQUIRED_CAPTURE_CHANGE# | NUMBER | Minimum REQUIRED_CHECKPOINT_SCN for all local capture processes on the database |
| CDB | VARCHAR2(3) | Possible values are:<br>• YES if the database is a CDB<br>• NO if the database is not a CDB |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| PENDING_ROLE_CHANGE_TASKS | VARCHAR2(512) | Tasks remaining after an Oracle Data Guard role change. Possible values:<br>• NONE: No pending tasks remain<br>• NOT APPLICABLE: The database is either standby or is not a DG_CONFIG member<br>• BUILD_PENDING: The database was formerly a logical standby and has not yet taken a snapshot of its data dictionary into the redo stream<br>• SRL_ARCHIVE_PENDING: The database was formerly a logical standby and the standby redo logs associated with the earlier failover operation have not yet been archived<br>• ERROR: The database was formerly a logical standby and the snapshot of the dictionary failed<br>• UNKNOWN: The database is not open or the query failed |
| CON_DBID | NUMBER | The database ID of the PDB |
| FORCE_FULL_DB_CACHING[1] | VARCHAR2(3) | Indicates the status of the force full database caching feature in the database. Possible values:<br>• YES - The database is in force full database caching mode.<br>• NO - The database is not in force full database caching mode.<br>**See Also**: *Oracle Database SQL Language Reference* for information about the FORCE FULL DATABASE CACHING clause for the ALTER DATABASE statement |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"DBA_SUPPLEMENTAL_LOGGING" for more information about supplemental logging in a PDB

# V$DATABASE_BLOCK_CORRUPTION

V$DATABASE_BLOCK_CORRUPTION displays information about database blocks that were corrupted after the last backup.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Absolute file number of the data file that contains the corrupt blocks |
| BLOCK# | NUMBER | Block number of the first corrupt block in the range of corrupted blocks |

| Column | Datatype | Description |
|---|---|---|
| BLOCKS | NUMBER | Number of corrupted blocks found starting with BLOCK# |
| CORRUPTION_CHANGE# | NUMBER | Change number at which the logical corruption was detected. Set to 0 to indicate media corruption. |
| CORRUPTION_TYPE | VARCHAR2(9) | Type of block corruption in the data file:<br>• ALL ZERO - Block header on disk contained only zeros. The block may be valid if it was never filled and if it is in an Oracle7 file. The buffer will be reformatted to the Oracle8 standard for an empty block.<br>• FRACTURED - Block header looks reasonable, but the front and back of the block are different versions.<br>• CHECKSUM - optional check value shows that the block is not self-consistent. It is impossible to determine exactly why the check value fails, but it probably fails because sectors in the middle of the block are from different versions.<br>• CORRUPT - Block is wrongly identified or is not a data block (for example, the data block address is missing)<br>• LOGICAL - Block is logically corrupt |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$NONLOGGED_BLOCK" for information about nonlogged blocks

# V$DATABASE_INCARNATION

V$DATABASE_INCARNATION displays information about all database incarnations. Oracle creates a new incarnation whenever a database is opened with the RESETLOGS option. Records about the current and immediately previous incarnation are also contained in the V$DATABASE view.

| Column | Datatype | Description |
|---|---|---|
| INCARNATION# | NUMBER | Record ID for the branch record in the control file |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs system change number (SCN) for the incarnation of the current row |
| RESETLOGS_TIME | DATE | Resetlogs timestamp for the incarnation of the current row |
| PRIOR_RESETLOGS_CHANGE# | NUMBER | Resetlogs SCN for the previous incarnation |
| PRIOR_RESETLOGS_TIME | DATE | Resetlogs timestamp for the previous incarnation |

| Column | Datatype | Description |
|---|---|---|
| STATUS | VARCHAR2(7) | Incarnation status:<br>• ORPHAN - Orphan incarnation<br>• CURRENT - Current incarnation of the database<br>• PARENT - Parent of the current incarnation |
| RESETLOGS_ID | NUMBER | Branch ID for the incarnation of the current row (used by user-managed recovery/RMAN restore to get unique names for archived logs across incarnations) |
| PRIOR_INCARNATION# | NUMBER | Parent incarnation record ID if nonzero |
| FLASHBACK_DATABASE_ALLOWED | VARCHAR2(26) | Indicate whether or not Flashback Database can be performed into SCNs or timestamps in the incarnation. A value of YES means that you can flashback to some point in that incarnation. A value of NO indicates that you cannot flashback into the incarnation. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$DATABASE"

# V$DATAFILE

V$DATAFILE displays datafile information from the control file.

**See Also:**

"V$DATAFILE_HEADER", which displays information from data file headers

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | File identification number |
| CREATION_CHANGE# | NUMBER | Change number at which the data file was created |
| CREATION_TIME | DATE | Timestamp of the data file creation |
| TS# | NUMBER | Tablespace number |
| RFILE# | NUMBER | Tablespace relative data file number |
| STATUS | VARCHAR2(7) | Type of file (system or user) and its status. Values: OFFLINE, ONLINE, SYSTEM, RECOVER, SYSOFF (an offline file from the SYSTEM tablespace) |

| Column | Datatype | Description |
|---|---|---|
| ENABLED | VARCHAR2(10) | Describes how accessible the file is from SQL:<br>• DISABLED - No SQL access allowed<br>• READ ONLY - No SQL updates allowed<br>• READ WRITE - Full access allowed<br>• UNKNOWN - Unknown whether SQL updates would be allowed or not |
| CHECKPOINT_CHANGE# | NUMBER | SCN at last checkpoint |
| CHECKPOINT_TIME | DATE | Timestamp of the checkpoint# |
| UNRECOVERABLE_CHANGE# | NUMBER | Last unrecoverable change number made to this data file. If the database is in ARCHIVELOG mode, then this column is updated when an unrecoverable operation completes. If the database is not in ARCHIVELOG mode, this column does not get updated. |
| UNRECOVERABLE_TIME | DATE | Timestamp of the last unrecoverable change. This column is updated only if the database is in ARCHIVELOG mode. |
| LAST_CHANGE# | NUMBER | Last change number made to this data file (null if the data file is being changed) |
| LAST_TIME | DATE | Timestamp of the last change |
| OFFLINE_CHANGE# | NUMBER | Offline change number of the last offline range. This column is updated only when the data file is brought online. |
| ONLINE_CHANGE# | NUMBER | Online change number of the last offline range |
| ONLINE_TIME | DATE | Online timestamp of the last offline range |
| BYTES | NUMBER | Current data file size (in bytes); 0 if inaccessible |
| BLOCKS | NUMBER | Current data file size (in blocks); 0 if inaccessible |
| CREATE_BYTES | NUMBER | Size when created (in bytes) |
| BLOCK_SIZE | NUMBER | Block size of the data file |
| NAME | VARCHAR2(513) | Name of the data file |
| PLUGGED_IN | NUMBER | Describes whether the tablespace is plugged in. The value is 1 if the tablespace is plugged in and has not been made read/write, 0 if not. |
| BLOCK1_OFFSET | NUMBER | Offset from the beginning of the file to where the Oracle generic information begins. The exact length of the file can be computed as follows: BYTES + BLOCK1_OFFSET. |
| AUX_NAME | VARCHAR2(513) | Auxiliary name that has been set for this file via CONFIGURE AUXNAME |
| FIRST_NONLOGGED_SCN | NUMBER | First nonlogged SCN (check in standby database) |
| FIRST_NONLOGGED_TIME | DATE | First nonlogged time (check in standby database) |

| Column | Datatype | Description |
|---|---|---|
| FOREIGN_DBID | NUMBER | Foreign DBID from which this data file came from. The value is 0 if this file is not a foreign database file. |
| FOREIGN_CREATION_CHANGE# | NUMBER | Creation SCN of a foreign data file. The value is 0 if this file is not a foreign database file. |
| FOREIGN_CREATION_TIME | DATE | Creation time of a foreign data file. The value is 0 if this file is not a foreign database file. |
| PLUGGED_READONLY | VARCHAR2(3) | YES if this is a transported read-only foreign file; otherwise NO. |
| PLUGIN_CHANGE# | NUMBER | SCN at which the foreign data file was transported into the database. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_CHANGE# | NUMBER | The SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_TIME | DATE | The time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DATAFILE_COPY

V$DATAFILE_COPY displays data file copy information from the control file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Data file copy record ID |
| STAMP | NUMBER | Data file copy record stamp |
| NAME | VARCHAR2(513) | File name of the data file copy. The maximum length of the name is operating system dependent. |
| TAG | VARCHAR2(32) | Data file copy tag |
| FILE# | NUMBER | Absolute data file number |
| RFILE# | NUMBER | Tablespace relative data file number |
| CREATION_CHANGE# | NUMBER | Data file creation change# |
| CREATION_TIME | DATE | Data file creation timestamp |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change number of the data file when the copy was made |
| RESETLOGS_TIME | DATE | Resetlogs timestamp of the data file when the copy was made |

| Column | Datatype | Description |
|---|---|---|
| INCREMENTAL_LEVEL | NUMBER | Normal full backups have a NULL value, level 0 incremental backups have a value of 0, and level 1 incremental backups have a value of 1 |
| CHECKPOINT_CHANGE# | NUMBER | Checkpoint change number of the data file when the copy was made |
| CHECKPOINT_TIME | DATE | Checkpoint timestamp of the data file when the copy was made |
| ABSOLUTE_FUZZY_CHANGE# | NUMBER | Highest change seen when the data file was copied |
| RECOVERY_FUZZY_CHANGE# | NUMBER | Highest change written to the file by media recovery |
| RECOVERY_FUZZY_TIME | DATE | Timestamp of the highest change written to the file by media recovery |
| ONLINE_FUZZY | VARCHAR2(3) | (YES\|NO) If set to YES, this is a copy taken using an operating system utility after a crash or offline immediate (or an invalid copy taken while data file was online and the database open). Recovery will need to apply all redo up to the next crash recovery marker to make the file consistent. |
| BACKUP_FUZZY | VARCHAR2(3) | (YES\|NO) If set to YES, this is a copy taken using the BEGIN BACKUP/END BACKUP technique. Recovery will need to apply all redo up to the end backup marker to make this copy consistent. |
| MARKED_CORRUPT | NUMBER | Number of blocks marked corrupt by this copy operation. That is, blocks that were not marked corrupted in the source data file, but were detected and marked as corrupted during the copy operation. |
| MEDIA_CORRUPT | NUMBER | Total number of media corrupt blocks. For example, blocks with checksum errors are marked media corrupt. |
| LOGICALLY_CORRUPT | NUMBER | Total number of logically corrupt blocks. For example, applying redo for unrecoverable operations will mark affected blocks logically corrupt. |
| BLOCKS | NUMBER | Size of the data file copy in blocks (also the size of the data file when the copy was made) |
| BLOCK_SIZE | NUMBER | Block size of the data file |
| OLDEST_OFFLINE_RANGE | NUMBER | RECID of the oldest offline range record in this control file copy; 0 for data file copies |
| DELETED | VARCHAR2(3) | (YES\|NO) If set to YES the data file copy has been deleted or overwritten |
| STATUS | VARCHAR2(1) | Identifies the status of this data file copy. Possible values are:<br>A - Available<br>D - Deleted<br>U - Unavailable<br>X - Expired |
| COMPLETION_TIME | DATE | Time when the copy was completed |

| Column | Datatype | Description |
|---|---|---|
| CONTROLFILE_TYPE | VARCHAR2(1) | B indicates normal copies<br>S indicates standby copies |
| KEEP | VARCHAR2(3) | (YES/NO) Indicates whether or not this backup set has a retention policy that is different than the value for the configure retention policy |
| KEEP_UNTIL | DATE | If KEEP_UNTIL is specified, this is the date after which the backup becomes obsolete. If this column is null, then the backup never expires. |
| KEEP_OPTIONS | VARCHAR2(11) | Lists additional retention options for this backup set. Possible values are:<br>LOGS - The logs need to recover this backup are kept<br>NOLOGS - The logs needed to recover this backup will not be kept |
| SCANNED | VARCHAR2(3) | Indicates whether RMAN scanned the file (YES) or not (NO) |
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether the file was created in the fast recovery area (YES) or not (NO) |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS record stamp |
| CONVERTED_FILE | VARCHAR2(3) | (YES/NO) Indicates whether or not the data file copy was created using the RMAN CONVERT command |
| SAME_ENDIAN | VARCHAR2(3) | If the value of the CONVERTED_FILE column is YES, then this column indicates whether the data file copy has the same endianness as the source data file (YES) or not (NO); otherwise NULL |
| FOREIGN_DBID | NUMBER | Foreign DBID from which this data file came from. The value is 0 if this file is not a foreign database file. |
| PLUGGED_READONLY | VARCHAR2(3) | YES if this is a transported read-only foreign file; otherwise NO. |
| PLUGIN_CHANGE# | NUMBER | SCN at which the foreign data file was transported into the database. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_CHANGE# | NUMBER | The SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_TIME | DATE | The time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| BACKED_BY_VSS | VARCHAR2(3) | Whether or not the file has been backed up by Volume Shadow Copy Service (VSS). This column is reserved for internal use. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |
| BACKED_BY_PDB | VARCHAR2(3) | Recovery Manager (RMAN) allows a PDB to be backed up in two ways. The value in this column indicates how the PDB backup was taken:<br>• YES: The backup was taken when connected to the PDB<br>• NO: The backup was taken when connected to the root container |

# V$DATAFILE_HEADER

V$DATAFILE_HEADER displays data file information from the data file headers.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Data file number (from control file) |
| STATUS | VARCHAR2(7) | ONLINE \| OFFLINE (from control file) |
| ERROR | VARCHAR2(18) | NULL if the data file header read and validation were successful. If the read failed then the rest of the columns are NULL. If the validation failed then the rest of columns may display invalid data. If there is an error then usually the data file must be restored from a backup before it can be recovered or used. |
| FORMAT | NUMBER | Indicates the format for the header block. The possible values are 6, 7, 8, 10 or 0.<br>6 - indicates Oracle Version 6<br>7 - indicates Oracle Version 7<br>8 - indicates Oracle Version 8<br>10 - indicates Oracle Version 10<br>0 - indicates the format could not be determined (for example, the header could not be read) |
| RECOVER | VARCHAR2(3) | File needs media recovery (YES \| NO) |
| FUZZY | VARCHAR2(3) | File is fuzzy (YES \| NO) |
| CREATION_CHANGE# | NUMBER | Data file creation change# |
| CREATION_TIME | DATE | Data file creation timestamp |
| TABLESPACE_NAME | VARCHAR2(30) | Tablespace name |
| TS# | NUMBER | Tablespace number |
| RFILE# | NUMBER | Tablespace relative data file number |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change# |
| RESETLOGS_TIME | DATE | Resetlogs timestamp |
| CHECKPOINT_CHANGE# | NUMBER | Data file checkpoint change# |
| CHECKPOINT_TIME | DATE | Data file checkpoint timestamp |

| Column | Datatype | Description |
|---|---|---|
| CHECKPOINT_COUNT | NUMBER | Data file checkpoint count |
| BYTES | NUMBER | Current data file size in bytes |
| BLOCKS | NUMBER | Current data file size in blocks |
| NAME | VARCHAR2(513) | Data file name |
| SPACE_HEADER | VARCHAR2(40) | Represents the block address of a space file header block of a locally managed data file |
| LAST_DEALLOC_CHANGE# | VARCHAR2(16) | Last deallocated SCN |
| UNDO_OPT_CURRENT_CHANGE# | VARCHAR2(40) | For internal use only |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DATAGUARD_CONFIG

V$DATAGUARD_CONFIG displays the unique database names defined with the DB_UNIQUE_NAME and LOG_ARCHIVE_CONFIG initialization parameters, providing a view of the Oracle Data Guard environment from any database in the configuration.

The first row of the view lists the unique database name of the current database that was specified with the DB_UNIQUE_NAME initialization parameter. Additional rows reflect the unique database names of the other databases in the configuration that were specified with the DG_CONFIG keyword of the LOG_ARCHIVE_CONFIG initialization parameter.

| Column | Datatype | Description |
|---|---|---|
| DB_UNIQUE_NAME | VARCHAR2(30) | Unique database name |
| PARENT_DBUN | VARCHAR2(30) | DB_UNIQUE_NAME of the parent database, also known as the database that supplies live redo to the destination.<br>For example, suppose Boston is the DB_UNIQUE_NAME of the primary database, Chicago is the DB_UNIQUE_NAME of the Far Sync Instance, and Seattle is the DB_UNIQUE_NAME of the terminal standby database. Since Boston is the primary database, it has no parent database so the PARENT_DBUN for Boston will be NULL. Since Boston services Chicago, the PARENT_DBUN of Chicago will be Boston. Since Chicago services Seattle, Chicago will be the PARENT_DBUN for Seattle. |
| DEST_ROLE | VARCHAR2(17) | Type of archival destination database:<br>• LOCAL - Local to primary database<br>• PHYSICAL - Physical standby<br>• FAR SYNC INSTANCE - Far Sync Instance |

| Column | Datatype | Description |
|---|---|---|
| | | • CROSS-INSTANCE - An instance of the primary<br>• LOGICAL - Logical standby<br>• SNAPSHOT - Snapshot standby database<br>• DOWNSTREAM - Streams downstream capture database |
| CURRENT_SCN | NUMBER | The SCN up to which the specified database has applied redo |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "DB_UNIQUE_NAME"
- "LOG_ARCHIVE_CONFIG"

# V$DATAGUARD_STATS

V$DATAGUARD_STATS displays information about Oracle Data Guard metrics when queried on a standby database. No rows are returned when queried on a primary database.

| Column | Datatype | Description |
|---|---|---|
| SOURCE_DBID | NUMBER | Database ID of the protected source database |
| SOURCE_DB_UNIQUE_NAME | VARCHAR2(32) | DB unique name of the protected source database |
| NAME | VARCHAR2(32) | Name of the metric:<br>• APPLY FINISH TIME - An estimate of the time needed to apply all received, but unapplied redo from the primary database. If there are one or more redo gaps on the standby database, an estimate of the time needed to apply all received, but unapplied redo up to the end of the last archived redo log before the beginning of the earliest redo gap.<br>• APPLY LAG - Apply lag is a measure of the degree to which the data in a standby database lags behind the data in the primary database, due to delays in propagating and applying redo to the standby database. This value is relevent only to the applying instance.<br>• TRANSPORT LAG - Transport lag is a measure of the degree to which the transport of redo to the standby database lags behind the generation of redo on the primary database. If there are one or more redo gaps on the standby database, the transport lag is calculated as if no redo has been received after the beginning of the earliest redo gap. |

| Column | Datatype | Description |
|---|---|---|
| | | • ESTIMATED STARTUP TIME - An estimate of the time needed to start and open the database. |
| VALUE | VARCHAR2(64) | Value of the metric |
| UNIT | VARCHAR2(30) | Unit of measurement |
| TIME_COMPUTED | VARCHAR2(30) | Local time at the standby database when the metric was computed |
| DATUM_TIME | VARCHAR2(30) | Local time at the standby database when the datum used to compute the metric was received<br>The APPLY LAG and TRANSPORT LAG metrics are computed based on data that is periodically received from the primary database. An unchanging value in this column across multiple queries indicates that the standby database is not receiving data from the primary database. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DATAGUARD_STATUS

V$DATAGUARD_STATUS displays messages recently written to the alert log or server process trace files that concern physical standby databases or redo transport services for all standby database types.

| Column | Datatype | Description |
|---|---|---|
| FACILITY | VARCHAR2(24) | Facility that encountered the event:<br>• Crash Recovery<br>• Log Transport Services<br>• Log Apply Services<br>• Role Management Services<br>• Remote File Server<br>• Fetch Archive Log<br>• Data Guard<br>• Network Services |
| SEVERITY | VARCHAR2(13) | Severity of the event:<br>• Informational - Informational message<br>• Warning - Warning message<br>• Error - Indicates the process has failed<br>• Fatal<br>• Control - An expected change in state such as the start or completion of an archival, log recovery, or switchover operation |
| DEST_ID | NUMBER | Destination ID number to which the event pertains. If the event does not pertain to a particular destination, then the value is 0. |

| Column | Datatype | Description |
|---|---|---|
| MESSAGE_NUM | NUMBER | A chronologically increasing number giving each event a unique number |
| ERROR_CODE | NUMBER | Error ID pertaining to the event |
| CALLOUT | VARCHAR2(3) | Reserved for future use |
| TIMESTAMP | DATE | Message date |
| MESSAGE | VARCHAR2(256) | A text message describing the event |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DB_CACHE_ADVICE

V$DB_CACHE_ADVICE contains rows that predict the number of physical reads for the cache size corresponding to each row. The rows also compute a "physical read factor," which is the ratio of the number of estimated reads to the number of reads actually performed by the real buffer cache during the measurement interval.

**See Also:**

"DB_CACHE_ADVICE"

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Buffer pool identifier (ranges from 1 to 8) |
| NAME | VARCHAR2(20) | Buffer pool name |
| BLOCK_SIZE | NUMBER | Block size in bytes for buffers in this pool. Possible values: the standard block size, the power of 2 nonstandard block sizes, 2048, 4096, 8192, 16384, 32768. |
| ADVICE_STATUS | VARCHAR2(3) | Status of the advisory. ON indicates it is currently running; OFF indicates it is disabled (in this case the estimates are historical and calculated when last enabled). |
| SIZE_FOR_ESTIMATE | NUMBER | Cache size for prediction (in megabytes) |
| SIZE_FACTOR | NUMBER | Size factor with respect to the current cache size |
| BUFFERS_FOR_ESTIMATE | NUMBER | Cache size for prediction (in terms of buffers) |
| ESTD_PHYSICAL_READ_FACTOR | NUMBER | Physical read factor for this cache size, which is the ratio of the number of estimated physical reads to the number of reads in the |

| Column | Datatype | Description |
|---|---|---|
| | | real cache. If there are no physical reads in the real cache, the value of this column is null. |
| ESTD_PHYSICAL_READS | NUMBER | Estimated number of physical reads for this cache size |
| ESTD_PHYSICAL_READ_TIME | NUMBER | Estimated disk read time (in seconds) |
| ESTD_PCT_OF_DB_TIME_FOR_READS | NUMBER | Estimated disk time as a percentage of the total time |
| ESTD_CLUSTER_READS | NUMBER | Estimated total number of blocks foreground processes read from the global cache (Oracle Real Application Clusters only) |
| ESTD_CLUSTER_READ_TIME | NUMBER | Estimated total amount of time, in seconds, foreground processes read from global cache (Oracle Real Application Clusters only) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DB_OBJECT_CACHE

V$DB_OBJECT_CACHE displays database objects that are cached in the library cache. Objects include tables, indexes, clusters, synonym definitions, PL/SQL procedures and packages, and triggers.

| Column | Datatype | Description |
|---|---|---|
| OWNER | VARCHAR2(64) | Owner of the object |
| NAME | VARCHAR2(1000) | Name of the object |
| DB_LINK | VARCHAR2(64) | Database link name, if any |
| NAMESPACE | VARCHAR2(64) | Library cache namespace of the object: TABLE/PROCEDURE, BODY, TRIGGER, INDEX, CLUSTER, OBJECT |
| TYPE | VARCHAR2(64) | Type of the object: INDEX, TABLE, CLUSTER, VIEW, SET, SYNONYM, SEQUENCE, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, TRIGGER, CLASS, OBJECT, USER, DBLINK |
| SHARABLE_MEM | NUMBER | Amount of sharable memory in the shared pool consumed by the object |
| LOADS | NUMBER | Number of times the object has been loaded. This count also increases when an object has been invalidated. |
| EXECUTIONS | NUMBER | Not used<br>**See Also:** "V$SQLAREA" to see actual execution counts |

| Column | Datatype | Description |
|---|---|---|
| LOCKS | NUMBER | Number of users currently locking this object |
| PINS | NUMBER | Number of users currently pinning this object |
| KEPT | VARCHAR2(3) | (YES \| NO) Depends on whether this object has been "kept" (permanently pinned in memory) with the PL/SQL procedure DBMS_SHARED_POOL.KEEP |
| CHILD_LATCH | NUMBER | Child latch number that is protecting the object. This column is obsolete and maintained for backward compatibility. |
| INVALIDATIONS | NUMBER | Total number of times objects in the namespace were marked invalid because a dependent object was modified |
| HASH_VALUE | NUMBER | Hash value of the object |
| LOCK_MODE | VARCHAR2(9) | Current lock mode of the object |
| PIN_MODE | VARCHAR2(9) | Current pin mode of the object |
| STATUS | VARCHAR2(19) | Status of the object |
| TIMESTAMP | VARCHAR2(19) | Timestamp for the specification of the objec |
| PREVIOUS_TIMESTAMP | VARCHAR2(19) | Previous timestamp for the specification of the object |
| LOCKED_TOTAL | NUMBER | Total number of times the object has been locked |
| PINNED_TOTAL | NUMBER | Total number of times the object has been pinned |
| PROPERTY | VARCHAR2(80) | Property of the object. Possible values include HOT or HOTCOPY when the library cache hot copy feature is used using DBMS_SHARED_POOL.MARKHOT. |
| FULL_HASH_VALUE | VARCHAR2(32) | Full hash value of the object |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CON_NAME | VARCHAR2(64) | Container name of the object. The value of this column is NULL in non-CDBs. |
| ADDR | RAW(8) | Address of the handle for this object |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SHARED_POOL.KEEP procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SHARED_POOL.MARKHOT procedure

# V$DB_PIPES

V$DB_PIPES displays the pipes that are currently represented in the shared pool for this instance.

| Column | Datatype | Description |
|---|---|---|
| OWNERID | NUMBER | Owner ID of the owner (if this is a private pipe), else NULL |
| NAME | VARCHAR2(1000) | Name of the pipe (for example, SCOTT.PIPE) |
| TYPE | VARCHAR2(7) | Type of the pipe:<br>• PUBLIC<br>• PRIVATE |
| PIPE_SIZE | NUMBER | Amount of memory the pipe uses<br>**Note:** The value of this column may be larger than maxpipesize because of an internal algorithm. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CON_NAME | VARCHAR2(64) | Container name of the object. The value of this column is NULL in non-CDBs. |

# V$DB_TRANSPORTABLE_PLATFORM

V$DB_TRANSPORTABLE_PLATFORM displays all platforms to which the database can be transported using the RMAN CONVERT DATABASE command. The transportable database feature only supports transports of the same endian platform. Therefore, V$DB_TRANSPORTABLE_PLATFORM displays fewer rows than V$TRANSPORTABLE_PLATFORM.

| Column | Datatype | Description |
|---|---|---|
| PLATFORM_ID | NUMBER | Platform identification number |
| PLATFORM_NAME | VARCHAR2(101) | Platform name |
| ENDIAN_FORMAT | VARCHAR2(14) | Platform endian format:<br>• Big<br>• Little<br>• UNKNOWN FORMAT |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$TRANSPORTABLE_PLATFORM"

# V$DBFILE

V$DBFILE displays all data files making up the database. This view is retained for historical compatibility. Use of V$DATAFILE is recommended instead.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | File identifier |
| NAME | VARCHAR2(513) | Name of the file |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$DATAFILE"

# V$DBLINK

V$DBLINK describes all database links (links with IN_TRANSACTION = YES) opened by the session issuing the query on V$DBLINK. These database links must be committed or rolled back before being closed.

| Column | Datatype | Description |
|---|---|---|
| DB_LINK | VARCHAR2(128) | Name of the database link |
| OWNER_ID | NUMBER | Owner of the database link UID |
| LOGGED_ON | VARCHAR2(3) | Whether the database link is currently logged on |
| HETEROGENEOUS | VARCHAR2(3) | Whether the database link is heterogeneous |
| PROTOCOL | VARCHAR2(6) | Communication protocol for the database link |
| OPEN_CURSORS | NUMBER | Whether there are open cursors for the database link |
| IN_TRANSACTION | VARCHAR2(3) | Whether the database link is currently in a transaction |

| Column | Datatype | Description |
|---|---|---|
| UPDATE_SENT | VARCHAR2(3) | Whether there has been an update on the database link |
| COMMIT_POINT_STRENGTH | NUMBER | Commit point strength of the transactions on the database link |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DEAD_CLEANUP

V$DEAD_CLEANUP shows the dead processes and killed sessions present in the instance and their cleanup status.

| Column | Datatype | Description |
|---|---|---|
| TYPE | VARCHAR2(64) | Indicates whether a row contains a DEAD PROCESS or KILLED SESSION |
| PADDR | RAW(8) | Process pointer. Can be joined with V$PROCESS.<br>• For a killed session that has not been moved, this is the current owner of the session.<br>• For a killed session that has been moved but has not been acked, this is the owner of the session at the time of the kill (the one that still needs to provide an ack)<br>• For a killed session that has been moved and acked, this will be NULL.<br>• There can be multiple rows for the same process if it owns multiple killed sessions. |
| SADDR | RAW(8) | Session pointer. Can be joined with V$SESSION. If a dead process has multiple user sessions, this is the current user session. If it owns other killed processes, there will be multiple rows for this dead process. |
| STATE | VARCHAR2(64) | Cleanup state:<br>• UNSAFE TO ATTEMPT - Occurs for a killed session that has not been moved, so no cleanup can occur on it yet<br>• CLEANUP PENDING - Occurs for a dead process / killed session that can be cleaned up, but PMON has not yet made an attempt<br>• RESOURCES FREED - Occurs for a dead process / killed session where all children have been freed, but the process / killed session itself is not yet freed<br>• RESOURCES FREED - PENDING ACK - Occurs for a killed session where all children have been freed, but the session itself cannot be freed until the owner has acked it<br>• PARTIAL CLEANUP - Occurs if some of the children have been cleaned up |

| Column | Datatype | Description |
|---|---|---|
| DEAD_TIME | NUMBER | Time since the process was marked dead or the session was marked killed (in seconds) |
| CLEANUP_ATTEMPTS | NUMBER | Number of times PMON has attempted cleanup |
| LAST_ATTEMPT | NUMBER | How long ago the last cleanup attempt occurred (in seconds) |
| CLEANUP_TIME | NUMBER | Total amount of time PMON has spent on cleanup of the process/session (in seconds) |
| NUM_BLOCKED | NUMBER | Number of sessions blocked on cleanup of this session |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DELETED_OBJECT

V$DELETED_OBJECT displays information about deleted archived logs, data file copies and backup pieces from the control file. The only purpose of this view is to optimize the recovery catalog resync operation. When an archived log, data file copy, or backup piece is deleted, the corresponding record is marked deleted.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Deleted object record ID |
| STAMP | NUMBER | Deleted object record stamp |
| TYPE | VARCHAR2(26) | Identifies the type of deleted object:<br>• ARCHIVED LOG<br>• BACKUP PIECE<br>• DATAFILE COPY<br>• PROXY COPY<br>• BACKUP PIECE AVAILABLE<br>• BACKUP PIECE EXPIRED<br>• PROXY COPY AVAILABLE<br>• PROXY COPY EXPIRED<br>• BACKUP PIECE UNAVAILABLE<br>• PROXY COPY UNAVAILABLE<br>• DATAFILE COPY AVAILABLE<br>• DATAFILE COPY EXPIRED<br>• DATAFILE COPY UNAVAILABLE<br>• ARCHIVED LOG AVAILABLE<br>• ARCHIVED LOG EXPIRED<br>• ARCHIVED LOG UNAVAILABLE<br>• BACKUP SET KEEP OPTIONS<br>• BACKUP SET KEEP UNTIL<br>• PROXY COPY KEEP OPTIONS<br>• PROXY COPY KEEP UNTIL |

| Column | Datatype | Description |
|---|---|---|
| | | • DATAFILE COPY KEEP OPTIONS<br>• DATAFILE COPY KEEP UNTIL<br>• DATAFILE RENAME ON RESTORE |
| OBJECT_RECID | NUMBER | Record ID of the deleted object |
| OBJECT_STAMP | NUMBER | Record timestamp of the deleted object |
| OBJECT_DATA | NUMBER | Displays additional internal information related to this deleted object. For internal Oracle use only. |
| SET_STAMP | NUMBER | Set stamp of the deleted object |
| SET_COUNT | NUMBER | Set count of the deleted object |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DG_BROKER_CONFIG

V$DG_BROKER_CONFIG provides a summary of an Oracle Data Guard broker configuration. This is similar to the DGMGRL CLI's SHOW CONFIGURATION command. It provides a view of the entire Oracle Data Guard broker configuration from any database in the configuration.

| Column | Datatype | Description |
|---|---|---|
| DATABASE | VARCHAR2(512) | Database unique name |
| CONNECT_IDENTIFIER | VARCHAR2(512) | Net connect identifier used to reach the database |
| DATAGUARD_ROLE | VARCHAR2(27) | Oracle Data Guard role of the database:<br>• PRIMARY<br>• PHYSICAL_STANDBY<br>• LOGICAL_STANDBY<br>• SNAPSHOT_STANDBY<br>• FAR SYNC INSTANCE |
| REDO_SOURCE | VARCHAR(30) | The database unique name of the redo source |
| ENABLED | VARCHAR2(5) | TRUE or FALSE to denote whether or not the database is managed by Oracle Data Guard broker |
| STATUS | NUMBER | An Oracle error number denoting the database's current status |
| VERSION | VARCHAR2(30) | Version of the broker configuration |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DIAG_INFO

V$DIAG_INFO describes the state of Automatic Diagnostic Repository (ADR) functionality using NAME=VALUE pairs.

| Column | Datatype | Description |
|---|---|---|
| INST_ID | NUMBER | Instance ID |
| NAME | VARCHAR2(64) | Identifies a piece of data that reflects the state of ADR, such as whether it is enabled, where the directories and files are located, and how many ongoing issues (incidents and problems) there are.<br>Possible values include:<br>• Diag Enabled: Indicates whether ADR is enabled or not<br>• ADR Base through Health Monitor: Display different directories (ADR base, ADR home, and then subdirectories within the ADR home)<br>• Default Trace File: Specifies the current default trace file for the current process<br>• Active Problem Count and Active Incident Count: Specify how many problems/incidents there are in this ADR that *either* happened in the last 24 hours *or* have a piece of metadata set indicating that it is a persistent error (like a disk corruption) |
| VALUE | VARCHAR2(512) | Describes the current state of the piece of data identified in the NAME column |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DISPATCHER

V$DISPATCHER displays information about the dispatcher processes.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(4) | Name of the dispatcher process |
| NETWORK | VARCHAR2(1024) | Network address of the dispatcher |
| PADDR | RAW(4 \| 8) | Process address |

| Column | Datatype | Description |
|---|---|---|
| STATUS | VARCHAR2(16) | Status of the dispatcher:<br>• WAIT - Idle<br>• SEND - Sending a message<br>• RECEIVE - Receiving a message<br>• CONNECT - Establishing a connection<br>• DISCONNECT - Handling a disconnect request<br>• BREAK - Handling a break<br>• TERMINATE - In the process of terminating<br>• ACCEPT - Accepting connections (no further information available)<br>• REFUSE - Rejecting connections (no further information available) |
| ACCEPT | VARCHAR2(3) | Indicates whether the dispatcher is accepting new connections (YES) or not (NO) |
| MESSAGES | NUMBER | Number of messages processed by the dispatcher |
| BYTES | NUMBER | Size (in bytes) of messages processed by the dispatcher |
| BREAKS | NUMBER | Number of breaks occurring in the connection |
| OWNED | NUMBER | Number of circuits owned by the dispatcher |
| CREATED | NUMBER | Number of circuits created by the dispatcher |
| IDLE | NUMBER | Total idle time for the dispatcher (in hundredths of a second) |
| BUSY | NUMBER | Total busy time for the dispatcher (in hundredths of a second) |
| LISTENER | NUMBER | Most recent Oracle error number the dispatcher received from the listener |
| CONF_INDX | NUMBER | Zero-based index of the DISPATCHERS configuration used by the dispatcher |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DISPATCHER_CONFIG

V$DISPATCHER_CONFIG displays information about the dispatcher configurations and their attributes.

| Column | Datatype | Description |
|---|---|---|
| CONF_INDX | NUMBER | Zero-based index of the DISPATCHERS configuration |
| NETWORK | VARCHAR2(1024) | Network protocol or listening address of dispatchers (may be truncated) |

| Column | Datatype | Description |
|---|---|---|
| DISPATCHERS | NUMBER | Number of dispatchers to maintain for the configuration |
| CONNECTIONS | NUMBER | Maximum number of concurrent connections per dispatcher |
| SESSIONS | NUMBER | Maximum number of concurrent sessions per dispatcher |
| MULTIPLEX | VARCHAR2(4) | Indicates whether Session Multiplexing is on:<br>• IN<br>• OUT<br>• BOTH<br>• OFF |
| LISTENER | VARCHAR2(1200) | Listeners to register dispatchers with (may be truncated) |
| SERVICE | VARCHAR2(512) | Service names supported (may be truncated) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DISPATCHER_RATE

V$DISPATCHER_RATE displays rate statistics for a number of activities performed by the dispatcher processes. Collected samples have an activity-specific "time-to-live" (TTL_* columns). Statistics are reported over the following two types of time intervals:

- Current statistics (CUR_ columns)

  Current statistics use samples collected over the most recent time-to-live interval.

- Historical statistics (AVG_ and most of the MAX_ columns)

  Historical statistics make use of all samples that are no longer current.

At the time of collection, a sample is current. After the time-to-live has elapsed, the sample becomes historical. Each type of activity has a specific scale (represented by the SCALE_* columns) at which the statistics are reported.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(4) | Name of the dispatcher process |
| PADDR | RAW(4 \| 8) | Address of the dispatcher process |
| CUR_LOOP_RATE | NUMBER | Rate at which the dispatcher has been iterating through its dispatching loop, reported over the past TTL_LOOPS, in iterations/SCALE_LOOPS |
| CUR_EVENT_RATE | NUMBER | Rate at which the dispatcher has been processing dispatcher events, reported over the past TTL_LOOPS, in events/ |

| Column | Datatype | Description |
|---|---|---|
| | | SCALE_LOOPS. Such dispatcher events include network events and shared server requests. |
| CUR_EVENTS_PER_LOOP | NUMBER | Average number of events the dispatcher has been processing in each iteration through its dispatching loop, reported over the past TTL_LOOPS, in events/iteration |
| CUR_MSG_RATE | NUMBER | Rate at which the dispatcher has been relaying messages between clients and shared servers, reported over the past TTL_MSG, in messages/SCALE_MSG |
| CUR_SVR_BUF_RATE | NUMBER | Rate at which the dispatcher has been relaying buffers to shared servers, reported over the past TTL_SVR_BUF, in messages/ SCALE_SVR_BUF |
| CUR_SVR_BYTE_RATE | NUMBER | Rate at which the dispatcher has been relaying data to shared servers, reported over the past TTL_SVR_BUF, in bytes/ SCALE_SVR_BUF |
| CUR_SVR_BYTE_PER_BUF | NUMBER | Average number of data types in each buffer relayed to shared servers, reported over the past TTL_SVR_BUF, in bytes/buffer |
| CUR_CLT_BUF_RATE | NUMBER | Rate at which the dispatcher has been relaying buffers to clients, reported over the past TTL_CLT_BUF, in buffers/ SCALE_CLT_BUF |
| CUR_CLT_BYTE_RATE | NUMBER | Rate at which the dispatcher has been relaying data to clients, reported over the past TTL_CLT_BUF, in bytes/ SCALE_CLT_BUF |
| CUR_CLT_BYTE_PER_BUF | NUMBER | Average number of data bytes in each buffer relayed to clients, reported over the past TTL_CLT_BUF, in bytes/buffer |
| CUR_BUF_RATE | NUMBER | Rate at which the dispatcher has been relaying buffers to either clients or shared servers, reported over the past TTL_BUF, in bytes/SCALE_BUF |
| CUR_BYTE_RATE | NUMBER | Rate at which the dispatcher has been relaying data to either clients or shared servers, reported over the past TTL_BUF, in bytes/SCALE_BUF |
| CUR_BYTE_PER_BUF | NUMBER | Average number of data bytes in each buffer relayed to either clients or shared servers, reported over the past TTL_BUF, in bytes/buffer |
| CUR_IN_CONNECT_RATE | NUMBER | Rate at which the dispatcher has been accepting incoming client connections, reported over the past TTL_IN_CONNECT, in connections/SCALE_IN_CONNECT |
| CUR_OUT_CONNECT_RATE | NUMBER | Rate at which the dispatcher has been establishing outbound connections, reported over the past TTL_OUT_CONNECT, in connections/SCALE_OUT_CONNECT |
| CUR_RECONNECT_RATE | NUMBER | In a connection pooling setup, the rate at which clients have been reconnecting to the dispatcher, reported over the past TTL_RECONNECT, in reconnections/SCALE_RECONNECT |
| MAX_LOOP_RATE | NUMBER | Maximum rate at which the dispatcher has ever iterated through its dispatching loop, reported in iterations/SCALE_LOOPS, over the dispatcher's lifetime excluding the past TTL_LOOPS |

| Column | Datatype | Description |
|---|---|---|
| MAX_EVENT_RATE | NUMBER | Maximum rate at which the dispatcher has ever processed dispatcher events, reported in events/SCALE_LOOPS, over the dispatcher's lifetime excluding the past TTL_LOOPS |
| MAX_EVENTS_PER_LOOP | NUMBER | Maximum number of events the dispatcher has ever processed in one iteration through its dispatching loop, reported in events/iteration, over the dispatcher's lifetime |
| MAX_MSG_RATE | NUMBER | Maximum rate at which the dispatcher has ever relayed messages between clients and shared servers, reported in messages/SCALE_MSG, over the dispatcher's lifetime excluding the past TTL_MSG |
| MAX_SVR_BUF_RATE | NUMBER | Maximum rate at which the dispatcher has ever relayed buffers to shared servers, reported in buffers/SCALE_SVR_BUF, over the dispatcher's lifetime excluding the past TTL_SVR_BUF |
| MAX_SVR_BYTE_RATE | NUMBER | Maximum rate at which the dispatcher has ever relayed data to shared servers, reported in bytes/SCALE_SVR_BUF, over the dispatcher's lifetime excluding the past TTL_SVR_BUF |
| MAX_SVR_BYTE_PER_BUF | NUMBER | Maximum number of data bytes the dispatcher has ever relayed in one buffer to a client, reported in bytes/buffer, over the dispatcher's lifetime |
| MAX_CLT_BUF_RATE | NUMBER | Maximum rate at which the dispatcher has ever relayed buffers to either clients or shared servers, reported in buffers/SCALE_CLT_BUF, over the dispatcher's life time excluding the past TTL_CLT_BUF |
| MAX_CLT_BYTE_RATE | NUMBER | Maximum rate at which the dispatcher has ever relayed buffers to clients, reported in bytes/SCALE_CLT_BUF, over the dispatcher's lifetime excluding the last TTL_CLT_BUF |
| MAX_CLT_BYTE_PER_BUF | NUMBER | Maximum number of data bytes the dispatcher has ever relayed in one buffer to a client, reported in bytes/buffer, over the dispatcher's lifetime |
| MAX_BUF_RATE | NUMBER | Maximum rate at which the dispatcher has ever relayed buffers to either clients or shared servers, reported in buffers/SCALE_BUF, over the dispatcher's lifetime, excluding the past TTL_BUF |
| MAX_BYTE_RATE | NUMBER | Maximum rate at which the dispatcher has ever relayed data to either clients or shared servers, reported in bytes/SCALE_BUF, over the dispatcher's lifetime excluding the past TTL_BUF |
| MAX_BYTE_PER_BUF | NUMBER | Maximum number of data bytes the dispatcher has ever relayed in one buffer to either a client or a shared server, reported in bytes/buffer, over the dispatcher's lifetime |
| MAX_IN_CONNECT_RATE | NUMBER | Maximum rate at which the dispatcher has ever accepted incoming client connections, reported in connections/SCALE_IN_CONNECT, over the dispatcher's lifetime excluding the past TTL_IN_CONNECT |
| MAX_OUT_CONNECT_RATE | NUMBER | Maximum rate at which the dispatcher has ever established outbound connections, reported in connections/ |

| Column | Datatype | Description |
|---|---|---|
| | | SCALE_OUT_CONNECT, over the dispatcher's lifetime excluding the past TTL_OUT_CONNECT |
| MAX_RECONNECT_RATE | NUMBER | In a connection pooling setup, the maximum rate at which clients have ever reconnected to this dispatcher, reported in reconnections/SCALE_RECONNECT, over the dispatcher's lifetime excluding the past TTL_RECONNECT |
| AVG_LOOP_RATE | NUMBER | Historical average rate at which the dispatcher has iterated through its dispatching loop, reported in iterations/ SCALE_LOOPS, over the dispatcher's lifetime excluding the past TTL_LOOPS |
| AVG_EVENT_RATE | NUMBER | Historical average rate at which the dispatcher has processed dispatcher events, reported in events/SCALE_LOOPS, over the dispatcher's lifetime excluding the past TTL_LOOPS |
| AVG_EVENTS_PER_LOOP | NUMBER | Historical average number of events the dispatcher has processed in one iteration through its dispatching loop, reported in events/ iteration, over the dispatcher's lifetime excluding the past TTL_LOOPS |
| AVG_MSG_RATE | NUMBER | Historical average rate at which the dispatcher has relayed messages between clients and shared servers, reported in messages/SCALE_MSG, over the dispatcher's lifetime excluding the past TTL_MSG |
| AVG_SVR_BUF_RATE | NUMBER | Historical average rate at which the dispatcher has relayed buffers to shared servers, reported in buffers/SCALE_SVR_BUF, over the dispatcher's lifetime excluding the past TTL_SVR_BUF |
| AVG_SVR_BYTE_RATE | NUMBER | Historical average rate at which the dispatcher has relayed data to shared servers, reported in bytes/SCALE_SVR_BUF, over the dispatcher's lifetime excluding the past TTL_SVR_BUF |
| AVG_SVR_BYTE_PER_BUF | NUMBER | Historical average number of data bytes per buffer the dispatcher has relayed to shared servers, reported in bytes/buffer, over the dispatcher's lifetime excluding the past TTL_SVR_BUF |
| AVG_CLT_BUF_RATE | NUMBER | Historical average rate at which the dispatcher has relayed buffers to clients, reported in buffers/SCALE_CLT_BUF, over the dispatcher's lifetime excluding the past TTL_CLT_BUF |
| AVG_CLT_BYTE_RATE | NUMBER | Historical average rate at which the dispatcher has relayed data to clients, reported in bytes/SCALE_CLT_BUF, over the dispatcher's lifetime excluding the past TTL_CLT_BUF |
| AVG_CLT_BYTE_PER_BUF | NUMBER | Historical average number of data bytes per buffer the dispatcher has relayed to clients, reported in bytes/buffer, over the dispatcher's lifetime excluding the past TTL_CLT_BUF |
| AVG_BUF_RATE | NUMBER | Historical average rate at which the dispatcher has relayed buffers to either clients or shared servers, reported in buffers/ SCALE_BUF, over the dispatcher's lifetime excluding the past TTL_BUF |
| AVG_BYTE_RATE | NUMBER | Historical average rate at which the dispatcher has relayed data to either clients or shared servers, reported in bytes/SCALE_BUF, over the dispatcher's lifetime excluding the past TTL_BUF |

| Column | Datatype | Description |
|---|---|---|
| AVG_BYTE_PER_BUF | NUMBER | Historical average number of data bytes per buffer the dispatcher has relayed to either clients or shared servers, reported in bytes/buffer, over the dispatcher's lifetime excluding the past TTL_BUF |
| AVG_IN_CONNECT_RATE | NUMBER | Historical average rate at which the dispatcher has accepted incoming client connections, reported in connections/SCALE_IN_CONNECT, over the dispatcher's lifetime excluding the past TTL_IN_CONNECT |
| AVG_OUT_CONNECT_RATE | NUMBER | Historical average rate at which the dispatcher has established outbound connections, reported in connections/SCALE_OUT_CONNECT, over the dispatcher's lifetime excluding the past TTL_OUT_CONNECT |
| AVG_RECONNECT_RATE | NUMBER | In a connection pooling setup, the historical average rate at which clients have reconnected to this dispatcher, reported in reconnections/SCALE_RECONNECT, over the dispatcher's lifetime excluding the past TTL_RECONNECT |
| TTL_LOOPS | NUMBER | Time-to-live for "loops" samples, reported in hundredths of a second. Default is 10 minutes. |
| TTL_MSG | NUMBER | Time-to-live for "messages" samples, reported in hundredths of a second. Default is 10 seconds. |
| TTL_SVR_BUF | NUMBER | Time-to-live for "buffers to servers" samples, reported in hundredths of a second. Default is 1 second. |
| TTL_CLT_BUF | NUMBER | Time-to-live for "buffers to clients" samples, reported in hundredths of a second. Default is 1 second. |
| TTL_BUF | NUMBER | Time-to-live for "buffers to clients/servers" samples, reported in hundredths of a second. Default is 1 second. |
| TTL_IN_CONNECT | NUMBER | Time-to-live for "inbound connections" samples, reported in hundredths of a second. Default is 10 minutes. |
| TTL_OUT_CONNECT | NUMBER | Time-to-live for "outbound connections" samples, reported in hundredths of a second. Default is 10 minutes. |
| TTL_RECONNECT | NUMBER | Time-to-live for "reconnections" samples, reported in hundredths of a second. Default is 10 minutes. |
| SCALE_LOOPS | NUMBER | Scale for "loops" statistics, reported in hundredths of a second. Default is 1 minute. |
| SCALE_MSG | NUMBER | Scale for "messages" statistics, reported in hundredths of a second. Default is 1 second. |
| SCALE_SVR_BUF | NUMBER | Scale for "buffers to servers" statistics, reported in hundredths of a second. Default is 1/10 second. |
| SCALE_CLT_BUF | NUMBER | Scale for "buffers to clients" statistics, reported in hundredths of a second. Default is 1/10 second. |
| SCALE_BUF | NUMBER | Scale for "buffers to clients/servers" statistics, reported in hundredths of a second. Default is 1/10 second. |
| SCALE_IN_CONNECT | NUMBER | Scale for "inbound connections" statistics, reported in hundredths of a second. Default is 1 minute. |

| Column | Datatype | Description |
|---|---|---|
| SCALE_OUT_CONNECT | NUMBER | Scale for "outbound connections" statistics, reported in hundredths of a second. Default is 1 minute. |
| SCALE_RECONNECT | NUMBER | Scale for "reconnections" statistics, reported in hundredths of a second. Default is 1 minute. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DNFS_CHANNELS

V$DNFS_CHANNELS displays information about the Oracle process connections (channels) open to NFS servers.

| Column | Datatype | Description |
|---|---|---|
| PNUM | NUMBER | Oracle process number |
| SVRNAME | VARCHAR2(255) | NFS server name |
| PATH | VARCHAR2(255) | Network path to the NFS server specified by IP address or by name |
| LOCAL | VARCHAR2(255) | Local path on the database host specified by IP address or by name |
| CH_ID | NUMBER | Direct NFS channel identifier |
| SVR_ID | NUMBER | Direct NFS server identifier |
| SENDS | NUMBER | Send operations over the channel since the last select |
| RECVS | NUMBER | Receive operations over the channel since the last select |
| PINGS | NUMBER | Ping operations over the channel since the last select |
| SPRECO | NUMBER | Reconnects for the channel on the same port since the last select, given that the reconnect occurred during the first series of reconnect attempts. If the first series of reconnect attempts fails and the reconnect process on that channel times out for 5 minutes, the next reconnect on that channel will not increment either the SPRECO or DPRECO columns, regardless of the port connected to. |
| DPRECO[1] | NUMBER | Reconnects for the channel on a different port since the last select, given that the reconnect occurred during the first series of reconnect attempts. If the first series of reconnect attempts fails and the reconnect process on that channel times out for 5 minutes, the next reconnect on that channel will not increment either the SPRECO or DPRECO columns, regardless of the port connected to. |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| RDMA | NUMBER | Indicates whether RDMA is enabled for the channel or not. Possible values:<br>• 0: RDMA is not enabled for the channel<br>• 1: RDMA is enabled for the channel |
| RDMA_CREDITS[1] | NUMBER | The number of RDMA credits supported by the server |
| CLIENTPORT[1] | NUMBER | The client port to which the channel is bounded |
| ACTIVE_SPEED[1] | NUMBER | The active speed of the HCA card present (in Gb/second) |
| PEAK_FMR[1] | NUMBER | The size of fast memory registration (FMR) memory registered by the channel (in bytes) |
| CURRENT_FMR[1] | NUMBER | The current FMR memory registered (in bytes) |
| FMRREG_COUNT[1] | NUMBER | The number of FMR memory registration calls for the channel |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

RDMA functionality is enabled only for the Exadata environment

# V$DNFS_FILES

V$DNFS_FILES displays information about the Oracle process files open through Direct NFS.

| Column | Datatype | Description |
|---|---|---|
| FILENAME | VARCHAR2(513) | File name |
| FILESIZE | NUMBER | File Size |
| PNUM | NUMBER | Oracle process number which opened the file |
| SVR_ID | NUMBER | Direct NFS server identifier which identifies the server the file is open on |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DNFS_SERVERS

V$DNFS_SERVERS displays information about the Direct NFS servers accessed by Direct NFS.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Direct NFS server identifier |
| SVRNAME | VARCHAR2(255) | NFS server name |
| DIRNAME | VARCHAR2(1024) | Mounted directory |
| MNTPORT | NUMBER | NFS mount port |
| NFSPORT | NUMBER | NFS port |
| NFSVERSION | VARCHAR2(8) | NFS version |
| WTMAX | NUMBER | WTMAX exported by the NFS server |
| RTMAX | NUMBER | RTMAX exported by the NFS server |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| RDMAENABLE | VARCHAR2(16) | Indicates whether the server is RDMA enabled. Possible values:<br>• YES: The server is RDMA enabled.<br>• NO: The server is not RDMA enabled. |
| RDMAPORT[1] | NUMBER | The port number for RDMA communications on the server. The value is always 20049. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

RDMA functionality is enabled only for the Exadata environment

# V$DNFS_STATS

V$DNFS_STATS displays information about the Oracle process NFS operation statistics issued by Direct NFS.

| Column | Datatype | Description |
|---|---|---|
| PNUM | NUMBER | Oracle process number that statistics are relevant to |
| NFS_NULL | NUMBER | Null |
| NFS_GETATTR | NUMBER | Get attributes |
| NFS_SETATTR | NUMBER | Set attributes |
| NFS_LOOKUP | NUMBER | Lookup object |
| NFS_ACCESS | NUMBER | Access object |
| NFS_READLINK | NUMBER | Read link |
| NFS_READ | NUMBER | Read file |
| NFS_WRITE | NUMBER | Write file |
| NFS_CREATE | NUMBER | Create file |
| NFS_MKDIR | NUMBER | Make directory |
| NFS_SYMLINK | NUMBER | Symbolic link |
| NFS_MKNOD | NUMBER | Make node |
| NFS_REMOVE | NUMBER | Remove file |
| NFS_RMDIR | NUMBER | Remove directory |
| NFS_RENAME | NUMBER | Rename |
| NFS_LINK | NUMBER | Link |
| NFS_READDIR | NUMBER | Read directory |
| NFS_READDIRPLUS | NUMBER | Read directory plus |
| NFS_FSSTAT | NUMBER | File system status |
| NFS_FSINFO | NUMBER | File system information |
| NFS_PATHCONF | NUMBER | Path configuration |
| NFS_COMMIT | NUMBER | Commit |
| NFS_MOUNT | NUMBER | Mount |
| NFS_READBYTES | NUMBER | Number of bytes read from NFS server |
| NFS_WRITEBYTES | NUMBER | Number of bytes written to NFS server |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$DYNAMIC_REMASTER_STATS

V$DYNAMIC_REMASTER_STATS displays statistical information about the dynamic remastering process of object affinity and read-mostly. All times are given in hundredths of a second, and total values reflect what has been collected since instance startup.

| Column | Datatype | Description |
|---|---|---|
| REMASTER_TYPE[1] | VARCHAR2(11) | Remaster process type. Possible values:<br>• AFFINITY: This value is used for the row containing statistics that pertain to dynamic remastering activity on object affinity.<br>• READ-MOSTLY: This value is used for the row containing statistics that pertain to dynamic remastering activity on read-mostly objects. |
| REMASTER_OPS | NUMBER | Total number of dynamic remastering operations |
| REMASTER_TIME | NUMBER | Total dynamic remastering time |
| REMASTERED_OBJECTS | NUMBER | Total number of objects dynamically remastered due to affinity |
| QUIESCE_TIME | NUMBER | Total quiesce step time |
| FREEZE_TIME | NUMBER | Total freeze step time |
| CLEANUP_TIME | NUMBER | Total cleanup step time |
| REPLAY_TIME | NUMBER | Total replay step time |
| FIXWRITE_TIME | NUMBER | Total fixwrite step time |
| SYNC_TIME | NUMBER | Total synchronization step time |
| RESOURCES_CLEANED | NUMBER | Total number of resources cleaned in the cleanup steps |
| REPLAYED_LOCKS_SENT | NUMBER | Total number of locks replayed to other instances in the replay steps |
| REPLAYED_LOCKS_RECEIVED | NUMBER | Total number of locks received from other instances in the replay steps |
| CURRENT_OBJECTS | NUMBER | Current number of objects remastered on this instance due to affinity |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$EDITIONABLE_TYPES

V$EDITIONABLE_TYPES lists all the editionable types based on the current compatibility setting. The SELECT privilege on V$EDITIONABLE_TYPES will be granted to PUBLIC.

The database compatibility setting will determine the set of editionable types.

With compatibility set to 11.2 or 12, this set includes FUNCTION, LIBRARY, PACKAGE, PACKAGE BODY, PROCEDURE, SYNONYM, TRIGGER, TYPE, TYPE BODY, and VIEW. With compatibility set to 12, the set will include these types as well as the SQL TRANSLATION PROFILE.

| Column | Datatype | Description |
|---|---|---|
| EDITIONABLE_TYPE | VARCHAR2(64) | The name of the type that is editionable |
| TYPE# | NUMBER | The number of the type that is editionable |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

For more information about edition-based redefinitions, see *Oracle Database Development Guide*.

# V$EMON

V$EMON displays performance statistics per event monitor (EMON) slave for diagnosability of notifications. All processing time and latency is in seconds.

| Column | Datatype | Description |
|---|---|---|
| EMON# | NUMBER | EMON identifier (0 - 9) |
| SID | NUMBER | Session identifier |
| STARTUP_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when this EMON slave was started |
| SERVER_TYPE | VARCHAR2(8) | Notification quality of the service provided by EMON:<br>• REGULAR<br>• RELIABLE |
| STATUS | VARCHAR2(6) | EMON status:<br>• IDLE<br>• ACTIVE |

| Column | Datatype | Description |
|---|---|---|
| STATUS_CHANGE_TIME | TIMESTAMP(3) WITH TIME ZONE | Time at which EMON switched to the current STATUS |
| NUM_NTFNS | NUMBER | Total number of notifications (including grouping notifications) |
| NUM_GROUPING_NTFNS | NUMBER | Number of grouping notifications |
| NUM_NTFNS_ALL_GROUPS | NUMBER | Total number of events in all notification groups |
| NUM_OCI_NTFNS | NUMBER | Number of OCI notifications |
| NUM_PLSQL_NTFNS | NUMBER | Number of PL/SQL notifications |
| NUM_EMAIL_NTFNS | NUMBER | Number of E-mail notifications |
| NUM_HTTP_NTFNS | NUMBER | Number of HTTP notifications |
| NUM_EVENTS_PROCESSED | NUMBER | Number of events posted by a publisher for which notifications have been delivered |
| NUM_EVENTS_PENDING | NUMBER | Number of events posted by a publisher for which notifications are not yet delivered |
| NUM_ANONYMOUS_NTFNS | NUMBER | Number of anonymous notifications |
| NUM_AQ_NTFNS | NUMBER | Number of AQ notifications |
| NUM_DBCHANGE_NTFNS | NUMBER | Number of DBChange notifications |
| TOTAL_ANONYMOUS_NTFN_TIME | NUMBER | Total time to process Anonymous notifications |
| TOTAL_AQ_NTFN_TIME | NUMBER | Total time to process AQ notifications |
| TOTAL_DBCHANGE_NTFN_TIME | NUMBER | Total time to process dbchange notifications |
| TOTAL_PLSQL_NTFN_TIME | NUMBER | Total time to process PL/SQL notifications |
| TOTAL_OCI_NTFN_TIME | NUMBER | Total time to process OCI notifications |
| TOTAL_EMAIL_NTFN_TIME | NUMBER | Total time to process E-mail notifications |
| TOTAL_HTTP_NTFN_TIME | NUMBER | Total time to process HTTP notifications |
| TOTAL_EMON_LATENCY | NUMBER | Total latency in processing events |
| REGISTRATIONS_EXPIRED | NUMBER | Number of expired registrations |
| REGISTRATIONS_PURGED | NUMBER | Number of purged registrations |

| Column | Datatype | Description |
|---|---|---|
| REGISTRATIONS_INVALID | NUMBER | Number of registrations invalidated due to notification delivery failure |
| LAST_UPDATE_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when statistics were last updated |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

# V$EMX_USAGE_STATS

V$EMX_USAGE_STATS is used to track how often each report in Oracle Enterprise Manager Database Express (EM Express) is used and how long the EM Express servlet takes to serve these reports to the client.

This view contains statistics such as the number of requests captured in the EM Express servlet for each report, total elapsed time for the EM Express servlet to render each report, as well as a detailed time breakdown including database login time, request initialization time, time to run the SQL query, and time to stream and send the query result back to the client.

All statistics are accumulated over all requests for each EM Express report since the last time the database instance was restarted.

It also includes the timestamp of the last request for each report.

| Column | Datatype | Description |
|---|---|---|
| REPORT | VARCHAR2(400) | Name of the EM Express report. All EM Express reports are XML reports with this format: %/orarep/*component*/*report* |
| COUNT | NUMBER | Number of requests captured by the EM Express servlet for this report since the last time the database instance was restarted |
| LOGIN_ELAPSED_TIME | NUMBER | Accumulated elapsed time for logging into the database across all requests for this report since the last time the database instance was restarted |
| INITREQ_ELAPSED_TIME | NUMBER | Accumulated elapsed time for initializing and preparing requests across all requests for this report since the last time the database instance was restarted |
| SQL_ELAPSED_TIME | NUMBER | Accumulated elapsed time for all requests that ran SQL queries for this report since the last time the database instance was restarted |
| SEND_ELAPSED_TIME | NUMBER | Accumulated elapsed time for streaming and sending the query result back to the client browser across all requests for this report since the last time the database instance was restarted |
| TOTAL_ELAPSED_TIME | NUMBER | Accumulated total elapsed time across all requests for this report since the last time the database instance was restarted |
| LAST_REQ_TIME | DATE | Timestamp of the last request for this report since the last time the database instance was restarted |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$ENABLEDPRIVS

V$ENABLEDPRIVS displays the system privileges that have been granted to the current user and the currently enabled roles. These privileges can be found in the table SYSTEM_PRIVILEGE_MAP.

| Column | Datatype | Description |
|---|---|---|
| PRIV_NUMBER | NUMBER | Numeric identifier of the system privilege |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"SYSTEM_PRIVILEGE_MAP"

# V$ENCRYPTED_TABLESPACES

V$ENCRYPTED_TABLESPACES displays information about the tablespaces that are encrypted.

| Column | Datatype | Description |
|---|---|---|
| TS# | NUMBER | Tablespace number |
| ENCRYPTIONALG | VARCHAR2(7) | Encryption algorithm:<br>• NONE<br>• 3DES168<br>• AES128<br>• AES192<br>• AES256 |

| Column | Datatype | Description |
|---|---|---|
| ENCRYPTEDTS | VARCHAR2(3) | Indicates whether the tablespace is encrypted (YES) or not (NO) |
| ENCRYTPEDKEY | RAW(32) | Encrypted version of the tablespace key for the encrypted tablespace |
| MASTERKEYID | RAW(16) | ID of the master key that was used to encrypt the tablespace key |
| BLOCKS_ENCRYPTED | NUMBER | Number of tablespace blocks that have been encrypted during the lifetime of this instance |
| BLOCKS_DECRYPTED | NUMBER | Number of tablespace blocks that have been decrypted during the lifetime of this instance |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ENCRYPTION_KEYS

V$ENCRYPTION_KEYS displays master key description attributes.

| Column | Datatype | Description |
|---|---|---|
| KEY_ID | VARCHAR2(78) | Master key identifier |
| TAG | VARCHAR2(4000) | Associated user-defined Information with the master key |
| CREATION_TIME | TIMESTAMP(6) WITH TIME ZONE | Time that the master key was created |
| ACTIVATION_TIME | TIMESTAMP(6) WITH TIME ZONE | Time that the master key was put into use |
| CREATOR | VARCHAR2(30) | User that created the master key |
| CREATOR_ID | NUMBER | User ID that created the master key |
| USER | VARCHAR2(30) | User that activated the master key |
| USER_ID | NUMBER | User ID that activated the master key |
| KEY_USE | VARCHAR2(10) | Indicates whether the master key is used for TDE operations in a PDB or not |
| KEYSTORE_TYPE | VARCHAR2(17) | Master key is in:<br>• HSM - Hardware Security Module<br>• SOFTWARE KEYSTORE<br>• UNDEFINED - This value is shown if the keystore has no information about the type of keystore where the master key resides |

| Column | Datatype | Description |
|---|---|---|
| ORIGIN | VARCHAR2(41) | Provides information about the origin of the master key:<br>• 1 - Was created locally in this database<br>• 2 - Was imported from another database<br>• 3 - Was imported but the key metadata was created locally due to activation<br>• 4 - Unknown if the key was imported or created locally but the key metadata was created locally due to activation<br>• 5 - Status of the master key is unknown. |
| BACKED_UP | VARCHAR2(9) | Indicates whether the key has been backed up or not |
| CREATOR_DBNAME | VARCHAR2(30) | Database that created the key |
| CREATOR_DBID | NUMBER | Database ID where the key was created |
| CREATOR_INSTANCE_NAME | VARCHAR2(30) | Instance name of the instance where the key was created |
| CREATOR_INSTANCE_NUMBER | NUMBER | Instance number of the instance where the key was created |
| CREATOR_INSTANCE_SERIAL | NUMBER | Serial number of the instance where the key was created |
| CREATOR_PDBNAME | VARCHAR2(30) | PDB where the key was created |
| CREATOR_PDBID | NUMBER | PDB ID where the key was created |
| CREATOR_PDBUID | NUMBER | PDB UID where the key was created |
| CREATOR_PDBGUID | RAW(16) | PDB GUID where the key was created |
| ACTIVATING_DBNAME | VARCHAR2(30) | Database that activated the key |
| ACTIVATING_DBID | NUMBER | Database ID where the key was activated |
| ACTIVATING_INSTANCE_NAME | VARCHAR2(30) | Instance name of the instance where the key was activated |
| ACTIVATING_INSTANCE_NUMBER | NUMBER | Instance number of the instance where the key was activated |
| ACTIVATING_INSTANCE_SERIAL | NUMBER | Serial number of the instance where the key was activated |
| ACTIVATING_PDBNAME | VARCHAR2(30) | PDB where the key was activated |
| ACTIVATING_PDBID | NUMBER | PDB ID where the key was activated |
| ACTIVATING_PDBUID | NUMBER | PDB UID where the key was activated |
| ACTIVATING_PDBGUID | RAW(16) | PDB GUID where the key was activated |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Advanced Security Guide* for information about keystore management

# V$ENCRYPTION_WALLET

V$ENCRYPTION_WALLET displays information on the status of the wallet and the wallet location for Transparent Data Encryption.

| Column | Datatype | Description |
|---|---|---|
| WRL_TYPE | VARCHAR2(20) | Type of the wallet resource locator (for example, FILE) |
| WRL_PARAMETER | VARCHAR2(4000) | Parameter of the wallet resource locator (for example, absolute file name if WRL_TYPE = FILE) |
| STATUS | VARCHAR2(30) | Status of the wallet:<br>• CLOSED<br>• NOT_AVAILABLE<br>• OPEN<br>• OPEN_NO_MASTER_KEY<br>• OPEN_UNKNOWN_MASTER_KEY_STATUS |
| WALLET_TYPE | VARCHAR2(20) | Displays the type of keystore being used, HSM or SOFTWARE_KEYSTORE.<br>If the keystore was created with the mkstore utility, then the WALLET_TYPE is UNKNOWN. |
| WALLET_ORDER | VARCHAR2(9) | If only a single wallet is configured, the value in this column is SINGLE.<br>When more than one wallet is configured, the value in this column shows whether the wallet is primary (holds the current master key) or secondary (holds old keys). The lookup of the master key will happen in the primary keystore first, and then in the secondary keystore, if required.<br>Possible values:<br>• PRIMARY<br>• SECONDARY<br>If there is only one type of keystore (Hardware Security Module or Software Keystore) being used, then PRIMARY will appear.<br>If both types are used, then the value in this column shows the order in which each keystore will be looked up. |
| FULLY_BACKED_UP | VARCHAR2(9) | Indicates whether all the keys in the keystore have been backed up |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ENQUEUE_LOCK

V$ENQUEUE_LOCK displays all locks owned by enqueue state objects. The columns in this view are identical to the columns in V$LOCK.

**See Also:**

"V$LOCK"

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of lock state object |
| KADDR | RAW(4 \| 8) | Address of lock |
| SID | NUMBER | Identifier for session holding or acquiring the lock |
| TYPE | VARCHAR2(2) | Type of lock. Lists user and system types that can have locks. |
| ID1 | NUMBER | Lock identifier #1 (depends on type) |
| ID2 | NUMBER | Lock identifier #2 (depends on type) |
| LMODE | NUMBER | Lock mode in which the session holds the lock:<br>• 0 - none<br>• 1 - null (NULL)<br>• 2 - row-S (SS)<br>• 3 - row-X (SX)<br>• 4 - share (S)<br>• 5 - S/Row-X (SSX)<br>• 6 - exclusive (X) |
| REQUEST | NUMBER | Lock mode in which the process requests the lock:<br>• 0 - none<br>• 1 - null (NULL)<br>• 2 - row-S (SS)<br>• 3 - row-X (SX)<br>• 4 - share (S)<br>• 5 - S/Row-X (SSX)<br>• 6 - exclusive (X) |
| CTIME | NUMBER | Time since current mode was granted |
| BLOCK | NUMBER | The lock is blocking another lock |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ENQUEUE_STAT

V$ENQUEUE_STAT displays statistics on the number of enqueue (lock) requests for each type of lock.

| Column | Datatype | Description |
|---|---|---|
| INST_ID | NUMBER | ID of the instance |
| EQ_TYPE | VARCHAR2(2) | Type of enqueue requested |
| TOTAL_REQ# | NUMBER | Total number of enqueue requests or enqueue conversions for this type of enqueue |
| TOTAL_WAIT# | NUMBER | Total number of times an enqueue request or conversion resulted in a wait |
| SUCC_REQ# | NUMBER | Number of times an enqueue request or conversion was granted |
| FAILED_REQ# | NUMBER | Number of times an enqueue request or conversion failed |
| CUM_WAIT_TIME | NUMBER | Total amount of time (in milliseconds) spent waiting for the enqueue or enqueue conversion |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ENQUEUE_STATISTICS

V$ENQUEUE_STATISTICS displays statistics on the number of enqueue (lock) requests for each type of lock. V$ENQUEUE_STATISTICS encompasses V$ENQUEUE_STAT and gives more detailed information (several rows for same enqueues with different reasons).

| Column | Datatype | Description |
|---|---|---|
| EQ_NAME | VARCHAR2(64) | Name of the enqueue request |
| EQ_TYPE | VARCHAR2(2) | Type of enqueue requested |
| REQ_REASON | VARCHAR2(64) | Reason for the enqueue request |
| TOTAL_REQ# | NUMBER | Total number of enqueue requests or enqueue conversions for this type of enqueue |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_WAIT# | NUMBER | Total number of times an enqueue request or conversion resulted in a wait |
| SUCC_REQ# | NUMBER | Number of times an enqueue request or conversion was granted |
| FAILED_REQ# | NUMBER | Number of times an enqueue request or conversion failed |
| CUM_WAIT_TIME | NUMBER | Total amount of time (in milliseconds) spent waiting for the enqueue or enqueue conversion |
| REQ_DESCRIPTION | VARCHAR2(4000) | Description of the enqueue request |
| EVENT# | NUMBER | Event number |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$ENQUEUE_STAT"

# V$EVENT_HISTOGRAM

V$EVENT_HISTOGRAM displays a histogram of the number of waits, the maximum wait, and total wait time on an event basis, in milliseconds. The histogram has buckets of time intervals from < 1 ms, < 2 ms, < 4 ms, < 8 ms, ... < $2^{21}$ ms, < $2^{22}$ ms, and >= $2^{22}$ ms.

The histogram will not be filled unless the TIMED_STATISTICS initialization parameter is set to true.

| Column | Datatype | Description |
|---|---|---|
| EVENT# | NUMBER | Event number |
| EVENT | VARCHAR2(64) | Name of the event |
| WAIT_TIME_MILLI | NUMBER | Amount of time the bucket represents (in milliseconds). If the duration = *num*, then this column represents waits of duration < *num* that are not included in any smaller bucket. |
| WAIT_COUNT | NUMBER | Number of waits of the duration belonging to the bucket of the histogram |
| LAST_UPDATE_TIME | VARCHAR2(64) | Indicates the last time the bucket was updated (the ending timestamp of the last wait falling into the bucket's duration) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"TIMED_STATISTICS"

# V$EVENT_HISTOGRAM_MICRO

V$EVENT_HISTOGRAM_MICRO displays a histogram of the number of waits, the maximum wait, and total wait time on an event basis, in microseconds. The histogram has buckets of time intervals from < 1 us, < 2 us, < 4 us, < 8 us, ... < $2^{31}$ us, < $2^{32}$ us, and >= $2^{32}$ us.

The histogram will not be filled unless the TIMED_STATISTICS initialization parameter is set to true.

| Column | Datatype | Description |
|---|---|---|
| EVENT# | NUMBER | Event number |
| EVENT | VARCHAR2(64) | Name of the event |
| WAIT_TIME_FORMAT | VARCHAR2(30) | A human readable time string which is converted from WAIT_TIME_MICRO. When WAIT_TIME_MICRO < 1 millisecond, WAIT_TIME_FORMAT is shown in microseconds. When WAIT_TIME_MICRO < 1 second, WAIT_TIME_FORMAT is shown in milliseconds. When WAIT_TIME_MICRO < 1 minute, WAIT_TIME_FORMAT is shown in seconds. When WAIT_TIME_MICRO > 1 minute, WAIT_TIME_FORMAT is shown in minutes and seconds. |
| WAIT_TIME_MICRO | NUMBER | Amount of time the bucket represents (in microseconds). If the duration = num, then this column represents waits of duration < num that are not included in any smaller bucket. |
| WAIT_COUNT | NUMBER | Number of waits of the duration belonging to the bucket of the histogram |
| LAST_UPDATE_TIME | VARCHAR2(64) | Indicates the last time the bucket was updated (the ending timestamp of the last wait falling into the bucket's duration) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$EVENT_NAME

V$EVENT_NAME displays information about wait events.

| Column | Datatype | Description |
|---|---|---|
| EVENT# | NUMBER | Number of the wait event |
| EVENT_ID | NUMBER | Identifier of the wait event |
| NAME | VARCHAR2(64) | Name of the wait event. Names that appear in this column remain stable across Oracle Database releases, and they can be relied on by customer scripts. |
| PARAMETER1 | VARCHAR2(64) | Description of the first parameter for the wait event |
| PARAMETER2 | VARCHAR2(64) | Description of the second parameter for the wait event |
| PARAMETER3 | VARCHAR2(64) | Description of the third parameter for the wait event |
| WAIT_CLASS_ID | NUMBER | Identifier of the class of the wait event |
| WAIT_CLASS# | NUMBER | Number of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | Name of the class of the wait event<br>**See Also:** "Classes of Wait Events" for a description of the different wait event classes |
| DISPLAY_NAME | VARCHAR2(64) | A clearer and more descriptive name for the wait event that appears in the NAME column. Names that appear in the DISPLAY_NAME column can change across Oracle Database releases, therefore customer scripts should not rely on names that appear in the DISPLAY_NAME column across releases. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$EVENTMETRIC

V$EVENTMETRIC displays values of wait event metrics for the most recent 60-second interval.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |

| Column | Datatype | Description |
|---|---|---|
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| EVENT# | NUMBER | Number of the event |
| EVENT_ID | NUMBER | Identifier of the event |
| NUM_SESS_WAITING | NUMBER | Number of sessions waiting at the end of the interval |
| TIME_WAITED | NUMBER | Time waited (in hundredths of a second) |
| WAIT_COUNT | NUMBER | Number of times waited |
| TIME_WAITED_FG | NUMBER | Time waited (in hundredths of a second), from foreground sessions |
| WAIT_COUNT_FG | NUMBER | Number of times waited, from foreground sessions |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$EXECUTION

V$EXECUTION displays information on parallel execution.

| Column | Datatype | Description |
|---|---|---|
| PID | NUMBER | Session ID |
| DEPTH | NUMBER | The depth |
| FUNCTION | VARCHAR2(10) | Session serial number |
| TYPE | VARCHAR2(7) | Name of the OBJECT_NODE in plan table |
| NVALS | NUMBER | Elapsed time for OBJECT_NODE |
| VAL1 | NUMBER | The value for number 1 |
| VAL2 | NUMBER | The value for number 2 |
| SEQH | NUMBER | A sequence |
| SEQL | NUMBER | A sequence |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FALSE_PING

V$FALSE_PING is deprecated. The information that was provided in this view is now provided in the V$INSTANCE_CACHE_TRANSFER and V$SEGMENT_STATISTICS views.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Data file identifier number (to find the file name, query DBA_DATA_FILES or V$DBFILE) |
| BLOCK# | NUMBER | Block number |
| STATUS | VARCHAR2(10) | Status of the block:<br>• free - Not currently in use<br>• xcur - Exclusive<br>• scur - Shared current<br>• cr - Consistent read<br>• read - Being read from disk<br>• mrec - In media recovery mode<br>• irec - In instance recovery mode |
| XNC | NUMBER | Number of PCM lock conversions from Exclusive mode due to contention with another instance. This column is obsolete and maintained for backward compatibility. |
| FORCED_READS | NUMBER | Number of times the block had to be reread from the cache because another instance has forced it out of this instance's cache by requesting the lock on the block in exclusive mode |
| FORCED_WRITES | NUMBER | Number of times GCS had to write this block to cache because this instance had used the block and another instance had requested the lock on the block in a conflicting mode |
| NAME | VARCHAR2(128) | Name of the database object containing the block |
| PARTITION_NAME | VARCHAR2(128) | NULL for nonpartitioned objects |
| KIND | VARCHAR2(15) | Type of database object |
| OWNER# | NUMBER | Owner number |
| LOCK_ELEMENT_ADDR | RAW(4 \| 8) | Address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| LOCK_ELEMENT_NAME | NUMBER | Name of the lock that contains the PCM lock that is covering the buffer |
| CLASS# | NUMBER | Lock element class |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$INSTANCE_CACHE_TRANSFER"
- "V$SEGMENT_STATISTICS"

# V$FAST_START_SERVERS

V$FAST_START_SERVERS provides information about all the recovery slaves performing parallel transaction recovery.

| Column | Datatype | Description |
|---|---|---|
| STATE | VARCHAR2(11) | State of the server (IDLE or RECOVERING) |
| UNDOBLOCKSDONE | NUMBER | Number of undo blocks done so far |
| PID | NUMBER | Process ID |
| XID | RAW(8) | Transaction ID |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FAST_START_TRANSACTIONS

V$FAST_START_TRANSACTIONS displays information about the progress of the transactions that Oracle is recovering.

| Column | Datatype | Description |
|---|---|---|
| USN | NUMBER | Undo segment number of the transaction |
| SLT | NUMBER | Slot within the rollback segment |
| SEQ | NUMBER | Incarnation number of the slot |
| STATE | VARCHAR2(16) | State of the transaction (may be TO BE RECOVERED, RECOVERED, or RECOVERING) |
| UNDOBLOCKSDONE | NUMBER | Number of undo blocks completed on the transaction |
| UNDOBLOCKSTOTAL | NUMBER | Total number of undo blocks that need recovery |
| PID | NUMBER | ID of the current server it has been assigned to |
| CPUTIME | NUMBER | Time for which recovery has progressed (in seconds) |
| PARENTUSN | NUMBER | Undo segment number of the parent transaction in PDML |
| PARENTSLT | NUMBER | Slot of the parent transaction in PDML |

| Column | Datatype | Description |
|---|---|---|
| PARENTSEQ | NUMBER | Sequence number of the parent transaction in PDML |
| XID | RAW(8) | Transaction ID |
| PXID | RAW(8) | Parent transaction ID |
| RCVSERVERS | NUMBER | Number of servers used in the last recovery |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FILE_CACHE_TRANSFER

V$FILE_CACHE_TRANSFER is deprecated. The information that was provided in this view is now provided in the V$INSTANCE_CACHE_TRANSFER and V$SEGMENT_STATISTICS views.

| Column | Datatype | Description |
|---|---|---|
| FILE_NUMBER | NUMBER | Number of the data file |
| X_2_NULL | NUMBER | Number of blocks with Exclusive-to-NULL conversions; always 0 |
| X_2_NULL_FORCED_WRITE | NUMBER | Number of Exclusive-to-NULL forced writes; always 0 |
| X_2_NULL_FORCED_STALE | NUMBER | Number of Exclusive-to-NULL blocks converted to CR; always 0 |
| X_2_S | NUMBER | Number of blocks with Exclusive-to-Shared conversions; always 0 |
| X_2_S_FORCED_WRITE | NUMBER | Number of Exclusive-to-Shared forced writes; always 0 |
| S_2_NULL | NUMBER | Number of blocks with Shared-to-NULL conversions; always 0 |
| S_2_NULL_FORCED_STALE | NUMBER | Number of Shared-to-NULL blocks converted to CR; always 0 |
| RBR | NUMBER | Number of reuse blocks cross-instance calls; always 0 |
| RBR_FORCED_WRITE | NUMBER | Number of blocks written due to reuse blocks cross-instance calls; always 0 |
| RBR_FORCED_STALE | NUMBER | Number of blocks marked as flushed due to reuse blocks cross-instance calls; always 0 |
| NULL_2_X | NUMBER | Number of blocks with NULL-to-Exclusive conversions; always 0 |

| Column | Datatype | Description |
|---|---|---|
| S_2_X | NUMBER | Number of blocks with Shared-to-Exclusive conversions; always 0 |
| NULL_2_S | NUMBER | Number of blocks with NULL-to-Shared conversions; always 0 |
| CR_TRANSFERS | NUMBER | Number of CR blocks received; always 0 |
| CUR_TRANSFERS | NUMBER | Number of current blocks received; always 0 |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$INSTANCE_CACHE_TRANSFER"
- "V$SEGMENT_STATISTICS"

# V$FILE_HISTOGRAM

V$FILE_HISTOGRAM displays a histogram of all synchronous single block reads on a per-file basis (for data files). The histogram has buckets of time intervals from < 1 ms, < 2 ms, < 4 ms, < 8 ms, ... < $2^{21}$ ms, < $2^{22}$ ms, and >= $2^{22}$ ms.

The histogram will not be filled unless the STATISTICS_LEVEL initialization parameter is set to ALL.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | File number |
| SINGLEBLKRDTIM_MILLI | NUMBER | Amount of time the bucket represents (in milliseconds). If the duration = *num*, then this column represents waits of duration < *num* that are not included in any smaller bucket. |
| SINGLEBLKRDS | NUMBER | Number of waits of the duration belonging to the bucket of the histogram |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FILEMETRIC

V$FILEMETRIC displays values of file metrics for the most recent 10-minute interval. A history of the last one hour will be kept in the system.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| FILE_ID | NUMBER | File number |
| CREATION_TIME | NUMBER | Timestamp of the file creation |
| AVERAGE_READ_TIME | NUMBER | Average file read time (in hundredths of a second) |
| AVERAGE_WRITE_TIME | NUMBER | Average file write time (in hundredths of a second) |
| PHYSICAL_READS | NUMBER | Number of physical reads |
| PHYSICAL_WRITES | NUMBER | Number of physical writes |
| PHYSICAL_BLOCK_READS | NUMBER | Number of physical block reads |
| PHYSICAL_BLOCK_WRITES | NUMBER | Number of physical block writes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FILEMETRIC_HISTORY

V$FILEMETRIC_HISTORY displays values of file metrics for all intervals in the last one hour.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| FILE_ID | NUMBER | File number |
| CREATION_TIME | NUMBER | Timestamp of the file creation |
| AVERAGE_READ_TIME | NUMBER | Average file read time (in hundredths of a second) |

| Column | Datatype | Description |
|---|---|---|
| AVERAGE_WRITE_TIME | NUMBER | Average file write time (in hundredths of a second) |
| PHYSICAL_READS | NUMBER | Number of physical reads |
| PHYSICAL_WRITES | NUMBER | Number of physical writes |
| PHYSICAL_BLOCK_READS | NUMBER | Number of physical block reads |
| PHYSICAL_BLOCK_WRITES | NUMBER | Number of physical block writes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FILESPACE_USAGE

V$FILESPACE_USAGE summarizes space allocation information of each data file and temp file.

| Column | Datatype | Description |
|---|---|---|
| TABLESPACE_ID | NUMBER | ID of the tablespace to which the file belongs |
| RFNO | NUMBER | Relative file number of the file |
| ALLOCATED_SPACE | NUMBER | Total allocated space in the file |
| FILE_SIZE | NUMBER | Current file size |
| FILE_MAXSIZE | NUMBER | Maximum file size |
| CHANGESCN_BASE | NUMBER | SCN base of the last change to the file |
| CHANGESCN_WRAP | NUMBER | SCN wrap of the last change to the file |
| FLAG | NUMBER | Flags for file attributes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FILESTAT

V$FILESTAT displays the number of physical reads and writes done and the total number of single-block and multiblock I/Os done at file level. As of Oracle Database

10g Release 2 (10.2), this view also includes reads done by RMAN processes for backup operations.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Number of the file |
| PHYRDS | NUMBER | Number of physical reads done |
| PHYWRTS | NUMBER | Number of times DBWR is required to write |
| PHYBLKRD | NUMBER | Number of physical blocks read |
| OPTIMIZED_PHYBLKRD | NUMBER | Number of physical reads from Database Smart Flash Cache blocks |
| PHYBLKWRT | NUMBER | Number of blocks written to disk, which may be the same as PHYWRTS if all writes are single blocks |
| SINGLEBLKRDS | NUMBER | Number of single block reads |
| READTIM | NUMBER | Time (in hundredths of a second) spent doing reads if the TIMED_STATISTICS parameter is true; 0 if false |
| WRITETIM | NUMBER | Time (in hundredths of a second) spent doing writes if the TIMED_STATISTICS parameter is true; 0 if false |
| SINGLEBLKRDTIM | NUMBER | Cumulative single block read time (in hundredths of a second) |
| AVGIOTIM | NUMBER | Average time (in hundredths of a second) spent on I/O, if the TIMED_STATISTICS parameter is true; 0 if false |
| LSTIOTIM | NUMBER | Time (in hundredths of a second) spent doing the last I/O, if the TIMED_STATISTICS parameter is true; 0 if false |
| MINIOTIM | NUMBER | Minimum time (in hundredths of a second) spent on a single I/O, if the TIMED_STATISTICS parameter is true; 0 if false |
| MAXIORTM | NUMBER | Maximum time (in hundredths of a second) spent doing a single read, if the TIMED_STATISTICS parameter is true; 0 if false |
| MAXIOWTM | NUMBER | Maximum time (in hundredths of a second) spent doing a single write, if the TIMED_STATISTICS parameter is true; 0 if false |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FIXED_TABLE

V$FIXED_TABLE displays all dynamic performance tables, views, and derived tables in the database. Some V$ tables (for example, V$ROLLNAME) refer to real tables and are therefore not listed.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(30) | Name of the object |
| OBJECT_ID | NUMBER | Identifier of the fixed object |
| TYPE | VARCHAR2(5) | Object type (TABLE \| VIEW) |
| TABLE_NUM | NUMBER | Number that identifies the dynamic performance table if it is of type TABLE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FIXED_VIEW_DEFINITION

V$FIXED_VIEW_DEFINITION contains the definitions of all the fixed views (views beginning with V$). Use this table with caution. Oracle tries to keep the behavior of fixed views the same from release to release, but the definitions of the fixed views can change without notice. Use these definitions to optimize your queries by using indexed columns of the dynamic performance tables.

| Column | Datatype | Description |
|---|---|---|
| VIEW_NAME | VARCHAR2(30) | Name of the fixed view |
| VIEW_DEFINITION | VARCHAR2(4000) | Definition of the fixed view |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FLASHBACK_DATABASE_LOG

V$FLASHBACK_DATABASE_LOG displays information about the flashback data. Use this view to help estimate the amount of flashback space required for the current workload.

| Column | Datatype | Description |
|---|---|---|
| OLDEST_FLASHBACK_SCN | NUMBER | Lowest system change number (SCN) in the flashback data, for any incarnation |
| OLDEST_FLASHBACK_TIME | DATE | Time of the lowest SCN in the flashback data, for any incarnation |

| Column | Datatype | Description |
|---|---|---|
| RETENTION_TARGET | NUMBER | Target retention time (in minutes) |
| FLASHBACK_SIZE | NUMBER | Current size (in bytes) of the flashback data |
| ESTIMATED_FLASHBACK_SIZE | NUMBER | Estimated size of flashback data needed (in bytes) for the current target retention |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FLASHBACK_DATABASE_LOGFILE

V$FLASHBACK_DATABASE_LOGFILE displays information about the flashback log files.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(513) | Name of the log file |
| LOG# | NUMBER | Log file number |
| THREAD# | NUMBER | Log file thread number |
| SEQUENCE# | NUMBER | Log file sequence number |
| BYTES | NUMBER | Log file size (in bytes) |
| FIRST_CHANGE# | NUMBER | Lowest system change number (SCN) in the log file |
| FIRST_TIME | DATE | Time of the first SCN in the log file |
| TYPE | VARCHAR2(9) | Log type:<br>• NORMAL<br>• RESERVED<br>• FREE<br>• TO DELETE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FLASHBACK_DATABASE_STAT

V$FLASHBACK_DATABASE_STAT displays statistics for monitoring the I/O overhead of logging flashback data. This view also displays the estimated flashback space needed based on previous workloads.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Beginning of the time interval |
| END_TIME | DATE | End of the time interval |
| FLASHBACK_DATA | NUMBER | Number of bytes of flashback data written during the interval |
| DB_DATA | NUMBER | Number of bytes of database data read and written during the interval |
| REDO_DATA | NUMBER | Number of bytes of redo data written during the interval |
| ESTIMATED_FLASHBACK_SIZE | NUMBER | Value of ESTIMATED_FLASHBACK_SIZE in V$FLASHBACK_DATABASE_LOG at the end of the time interval |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FLASHBACK_TXN_GRAPH

V$FLASHBACK_TXN_GRAPH displays a tabular representation of the transaction dependency graph. For each dependency edge, there could be multiple rows, one for each conflicting operation..

This view is relevant AFTER a compensating transaction has been started through the DBMS_FLASHBACK.TRANSACTION_BACKOUT() set of functions, and is no longer relevant once the compensating transaction is either committed or rolled back. It also provides a tabular representation of the undo SQL that is not available through the CLOB XML construct in the DBA_FLASHBACK_TXN_REPORT view.

| Column | Datatype | Description |
|---|---|---|
| COMPENSATING_XID | RAW(8) | Transaction ID of the compensating transaction |
| COMPENSATING_TXN_NAME | VARCHAR2(255) | Name of the compensating transaction |
| XID | RAW(8) | Transaction ID of a relevant transaction found in memory |
| TXN_NAME | VARCHAR2(255) | Name of the transaction with XID as the transaction ID; NULL if none |
| PARENT_XID | RAW(8) | Parent transaction ID (for a PDML transaction) |
| INTERESTING | NUMBER | If the transaction is in the transaction dependency graph |

| Column | Datatype | Description |
|---|---|---|
| ORIGINAL | NUMBER | If the transaction is part of the input set provided |
| BACKOUT_SEQ | NUMBER | Order in which the transaction has been backed out |
| NUM_PREDS | NUMBER | Number of predecessors of the transaction specified by XID in the transaction graph |
| NUM_SUCCS | NUMBER | Number of successors of the transaction specified by XID in the transaction graph |
| DEP_XID | RAW(8) | One dependent transaction ID of the transaction specified by XID. This is a particular child of XID. |
| DEP_TXN_NAME | VARCHAR2(255) | Transaction name, if any, for the transaction specified by DEP_XID |
| TXN_CONF_SQL_ID | NUMBER | SQL ID of undo SQL executed in the context of XID which conflicts with the dependent transaction |
| DEP_TXN_CONF_SQL_ID | NUMBER | SQL ID of undo SQL executed in the context of DEP_XID which conflicts with XID |
| CONFLICT_TYPE | VARCHAR2(32) | The type of conflict that the conflict resolution method is used to resolve: delete, uniqueness, or update |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "DBA_FLASHBACK_TXN_REPORT"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_FLASHBACK.TRANSACTION_BACKOUT procedures

# V$FLASHBACK_TXN_MODS

V$FLASHBACK_TXN_MODS displays the individual modifications of all the transactions in memory.

This view is relevant AFTER a compensating transaction has been started through the DBMS_FLASHBACK.TRANSACTION_BACKOUT() set of functions, and is no longer relevant once the compensating transaction is either committed or rolled back. It also provides a tabular representation of the undo SQL that is not available through the CLOB XML construct in the DBA_FLASHBACK_TXN_REPORT view.

| Column | Datatype | Description |
|---|---|---|
| COMPENSATING_XID | RAW(8) | Transaction ID of the compensating transaction |

| Column | Datatype | Description |
|---|---|---|
| COMPENSATING_TXN_NAME | VARCHAR2(255) | Name of the compensating transaction |
| XID | RAW(8) | Transaction ID of a relevant transaction found in memory |
| TXN_NAME | VARCHAR2(255) | Name of the transaction with XID as the transaction ID; NULL if none |
| PARENT_XID | RAW(8) | Parent transaction ID (for a PDML transaction) |
| INTERESTING | NUMBER | If the transaction is in the transaction dependency graph |
| ORIGINAL | NUMBER | If the transaction is part of the input set provided |
| BACKOUT_SEQ | NUMBER | Order in which the transaction has been backed out |
| UNDO_SQL | VARCHAR2(4000) | Undo SQL for the modification |
| UNDO_SQL_SQN | NUMBER | Order in which the given SQL has been executed to back out this transaction |
| UNDO_SQL_SUB_SQN | NUMBER | If the undo SQL is greater than 4000 bytes, then a sequence number, starting from 1, of a 4000-byte division of the undo SQL |
| BACKOUT_SQL_ID | NUMBER | SQL ID of the undo SQL (used only for this compensating transaction) |
| OPERATION | VARCHAR2(30) | Operation (such as insert/update/delete) performed by the forward-going operation |
| BACKEDOUT | NUMBER | Indicates whether the transaction has been backed out as of now |
| CONFLICT_MOD | NUMBER | If the concerned modification is causing a conflict |
| MODS_PER_LCR | NUMBER | Sometimes an LCR could cause multiple modifications (for example, an update of an IOT could actually be a delete followed by an insert) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "DBA_FLASHBACK_TXN_REPORT"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_FLASHBACK.TRANSACTION_BACKOUT procedures

# V$FLASHFILESTAT

V$FLASHFILESTAT displays statistics about Database Smart Flash Cache.

By taking snapshots of SINGLEBLKRDS and SINGLEBLKRDTIM_MICRO, you can easily calculate the average latency of all the flash files in a given time period

| Column | Datatype | Description |
|---|---|---|
| FLASHFILE# | NUMBER | The file number of the flash file |
| NAME | VARCHAR2(513) | Name and path of the flash file |
| BYTES | NUMBER | Size of the flash file, in bytes |
| ENABLED | NUMBER | Indicates whether this flash file is enabled or not |
| SINGLEBLKRDS | NUMBER | Number of reads to the flash file |
| SINGLEBLKRDTIM_MICRO | NUMBER | Cumulative latency of reading blocks from this particular flash file/device, in microseconds |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Administrator's Guide* for more information about configuring Database Smart Flash Cache

# V$FOREIGN_ARCHIVED_LOG

V$FOREIGN_ARCHIVED_LOG can be queried on a logical standby database to find out the list of foreign archived logs received by a database.

No rows are returned for this view on a physical standby database.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Archived log record ID |
| STAMP | NUMBER | Archived log record stamp |
| NAME | VARCHAR2(513) | Archived log file name. If set to NULL, either the log file was cleared before it was archived or an RMAN BACKUP command with the delete input option was executed to back up archivelog all (RMAN> backup archivelog all delete input;). |
| DEST_ID | NUMBER | Original destination from which the archive log was generated. The value is 0 if the destination identifier is not available |

| Column | Datatype | Description |
|---|---|---|
| THREAD# | NUMBER | Redo thread number |
| SEQUENCE# | NUMBER | Redo log sequence number |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change number of the database when the log was written |
| RESETLOGS_TIME | DATE | Resetlogs time of the database when the log was written |
| RESETLOGS_ID | NUMBER | Resetlogs identifier associated with the archived redo log |
| FIRST_CHANGE# | NUMBER | First change number in the archived log |
| FIRST_TIME | DATE | Timestamp of the first change |
| NEXT_CHANGE# | NUMBER | First change in the next log |
| NEXT_TIME | DATE | Timestamp of the next change |
| BLOCKS | NUMBER | Size of the archived log (in blocks) |
| BLOCK_SIZE | NUMBER | Redo log block size. This is the logical block size of the archived log, which is the same as the logical block size of the online log from which the archived log was copied. The online log logical block size is a platform-specific value that is not adjustable by the user. |
| CREATOR | VARCHAR2(7) | Creator of the archive log:<br>ARCH - Archiver process<br>FGRD - Foreground process<br>RMAN - Recovery Manager<br>SRMN - RMAN at standby<br>LGWR - Logwriter process |
| REGISTRAR | VARCHAR2(7) | Registrar of the entry:<br>RFS - Remote File Server process<br>ARCH - Archiver process<br>FGRD - Foreground process<br>RMAN - Recovery Manager<br>SRMN - RMAN at standby<br>LGWR - Logwriter process |
| ARCHIVED | VARCHAR2(3) | Indicates whether the online redo log was archived (YES) or whether RMAN only inspected the log and created a record for future application of redo logs during recovery (NO). |
| APPLIED | VARCHAR2(3) | Indicates whether the archivelog has been applied to its corresponding standby database (YES) or not (NO). The value is always NO for local destinations.<br>This column is meaningful at the standby site for the ARCHIVED_LOG entries with REGISTRAR='RFS' (which means this log is shipped from the primary to the standby database). If REGISTRAR='RFS' and APPLIED is NO, then the log has arrived at the standby but has not yet been applied. If REGISTRAR='RFS' and APPLIED is YES, the log has arrived and been applied at the standby database. |

| Column | Datatype | Description |
|---|---|---|
| | | You can use this field to identify archivelogs that can be backed up and removed from disk. |
| DELETED | VARCHAR2(3) | Indicates whether an RMAN DELETE command has physically deleted the archived log file from disk, as well as logically removing it from the control file of the target database and from the recovery catalog (YES) or not (NO) |
| STATUS | VARCHAR2(1) | Status of the archived log:<br>A - Available<br>D - Deleted<br>U - Unavailable<br>X - Expired |
| COMPLETION_TIME | DATE | Time when the archiving completed |
| DICTIONARY_BEGIN | VARCHAR2(3) | Indicates whether the log contains the start of a LogMiner dictionary (YES) or not (NO) |
| DICTIONARY_END | VARCHAR2(3) | Indicates whether the log contains the end of a LogMiner dictionary (YES) or not (NO) |
| END_OF_REDO | VARCHAR2(3) | Indicates whether the archived redo log contains the end of all redo information from the primary database () or not () |
| ARCHIVAL_THREAD# | NUMBER | Redo thread number of the instance that performed the archival operation. This column differs from the THREAD# column only when a closed thread is archived by another instance. |
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether the file was created in the fast recovery area (YES) or not (NO) |
| COMPRESSED | VARCHAR2(3) | Reserved for internal use |
| FAL | VARCHAR2(3) | Indicates whether the archive log was generated as the result of a FAL request (YES) or not (NO) |
| END_OF_REDO_TYPE | VARCHAR2(10) | Possible values are as follows:<br>SWITCHOVER - Shows archived redo log files that are produced at the end of a switchover<br>TERMINAL - Shows archived redo log files produced after a failover<br>RESETLOGS - Shows online redo log files archived on the primary database after an ALTER DATABASE OPEN RESETLOGS statement is issued<br>ACTIVATION - Shows any log files archived on a physical standby database after an ALTER DATABASE ACTIVATE STANDBY DATABASE statement is issued<br>"empty string" - Any empty string implies that the log is just a normal archival and was not archived due to any of the other events |
| SOURCE_DBID | NUMBER | Database ID of the source database that generated this archived log |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FS_FAILOVER_STATS

V$FS_FAILOVER_STATS displays statistics about Fast-Start Failovers (FSFOs) occurring on the system.

| Column | Datatype | Description |
|---|---|---|
| LAST_FAILOVER_TIME | VARCHAR2(20) | Timestamp of the last Fast-Start Failover |
| LAST_FAILOVER_REASON | VARCHAR2(255) | Reason for the last Fast-Start Failover |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$FS_OBSERVER_HISTOGRAM

V$FS_OBSERVER_HISTOGRAM displays statistics that are based on the frequency of successful pings between the observer and primary database for different time intervals. These statistics can be used to select an appropriate value for the FastStartFailoverThreshold configuration property for your environment.

| Column | Datatype | Description |
|---|---|---|
| WAIT_TIME | NUMBER | The time interval between a pair of successful observer pings (ping-pairs) to this instance. Note that the values in this column are the upper bound of the inter-ping interval samples represented by a given histogram bucket. If WAIT_TIME = *number*, then this column represents inter-ping intervals <= *number* that are not included in any smaller bucket. |
| WAIT_COUNT | NUMBER | The number of ping-pairs with an inter-ping interval that corresponds with this histogram bucket. |
| LAST_UPDATE_TIME | VARCHAR2(20) | The last time this row was updated |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Data Guard Broker* for more information about the `FastStartFailoverThreshold` configuration property

# V$GC_ELEMENT

`V$GC_ELEMENT` displays one entry for each global cache resource that is used by the buffer cache. The name of the global cache resource that corresponds to a lock element is {'BL', indx, class}. This is an Oracle Real Application Clusters view.

| Column | Datatype | Description |
|---|---|---|
| `GC_ELEMENT_ADDR` | `RAW(4 \| 8)` | Address of the lock element that contains the PCM lock that is covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| `INDX` | `NUMBER` | Platform specific lock manager identifier |
| `CLASS` | `NUMBER` | Platform specific lock manager identifier |
| `GC_ELEMENT_NAME` | `NUMBER` | Name of the lock that contains the PCM lock that is covering the buffer |
| `MODE_HELD` | `NUMBER` | Platform dependent value for lock mode held; often: 3 = share; 5 = exclusive |
| `BLOCK_COUNT` | `NUMBER` | Number of blocks covered by PCM lock |
| `RELEASING` | `NUMBER` | Nonzero if PCM lock is being downgraded |
| `ACQUIRING` | `NUMBER` | Nonzero if PCM lock is being upgraded |
| `WRITING` | `NUMBER` | If the GC_ELEMENT is being written, the write status |
| `RECOVERING` | `NUMBER` | If the GC_ELEMENT is being recovered, the recovery status |
| `LOCAL` | `NUMBER` | Zero if the GC_ELEMENT is local, one if it is global |
| `FLAGS` | `NUMBER` | Process level flags for the lock element |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GC_ELEMENTS_WITH_COLLISIONS

V$GC_ELEMENTS_WITH_COLLISIONS is deprecated. The information that was provided in this view is now provided in the V$INSTANCE_CACHE_TRANSFER and V$SEGMENT_STATISTICS views.

| Column | Datatype | Description |
|---|---|---|
| GC_ELEMENT_ADDR | RAW(4 \| 8) | Address of the lock element that contains the PCM lock covering the buffer. If more than one buffer has the same address, then these buffers are covered by the same PCM lock. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$INSTANCE_CACHE_TRANSFER"
- "V$SEGMENT_STATISTICS"

# V$GCSHVMASTER_INFO

V$GCSHVMASTER_INFO describes the current and previous master instances and the number of re-masterings of Global Cache Service resources except those belonging to files mapped to a particular master.

| Column | Datatype | Description |
|---|---|---|
| HV_ID | NUMBER | PCM hash value ID |
| CURRENT_MASTER | NUMBER | Master instance of this PCM hash value ID |
| PREVIOUS_MASTER | NUMBER | Previous master instance of this PCM hash value ID |
| REMASTER_CNT | NUMBER | Number of times this has been remastered |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GCSPFMASTER_INFO

V$GCSPFMASTER_INFO describes the current and previous master instances and the number of re-masterings of Global Cache Service resources belonging to files mapped to instances.

| Column | Datatype | Description |
|---|---|---|
| FILE_ID | NUMBER | File number |
| DATA_OBJECT_ID | NUMBER | Data object ID |
| GC_MASTERING_POLICY | VARCHAR2(11) | Data object type.<br>The possible values are Affinity or Read mostly. |
| CURRENT_MASTER | NUMBER | Master instance of this file |
| PREVIOUS_MASTER | NUMBER | Previous master instance of this file |
| REMASTER_CNT | NUMBER | Number of times this has been remastered |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GES_BLOCKING_ENQUEUE

V$GES_BLOCKING_ENQUEUE describes all locks currently known to lock manager that are being blocked or blocking others. The output of this view is a subset of the output from V$GES_ENQUEUE. This is an Oracle Real Application Clusters view.

**See Also:**

"V$GES_ENQUEUE" for a description of all locks known to the lock manager

| Column | Datatype | Description |
|---|---|---|
| HANDLE | RAW(4 \| 8) | Lock pointer |
| GRANT_LEVEL | VARCHAR2(9) | Granted level of the lock |
| REQUEST_LEVEL | VARCHAR2(9) | Requested level of the lock |
| RESOURCE_NAME1 | VARCHAR2(30) | Resource name for the lock |
| RESOURCE_NAME2 | VARCHAR2(30) | Resource name for the lock |
| PID | NUMBER | Process identifier which holds the lock |
| TRANSACTION_ID0 | NUMBER | Lower 4 bytes of the transaction identifier where the lock belongs to |

| Column | Datatype | Description |
|---|---|---|
| TRANSACTION_ID1 | NUMBER | Upper 4 bytes of the transaction identifier where the lock belongs to |
| GROUP_ID | NUMBER | Group identifier for the lock |
| OPEN_OPT_DEADLOCK | NUMBER | 1 if DEADLOCK open option is set, otherwise 0 |
| OPEN_OPT_PERSISTENT | NUMBER | 1 if PERSISTENT open option is set, otherwise 0 |
| OPEN_OPT_PROCESS_OWNED | NUMBER | 1 if PROCESS_OWNED open option is set, otherwise 0 |
| OPEN_OPT_NO_XID | NUMBER | 1 if NO_XID open option is set, otherwise 0 |
| CONVERT_OPT_GETVALUE | NUMBER | 1 if GETVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_PUTVALUE | NUMBER | 1 if PUTVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_NOVALUE | NUMBER | 1 if NOVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_DUBVALUE | NUMBER | 1 if DUBVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_NOQUEUE | NUMBER | 1 if NOQUEUE convert option is set, otherwise 0 |
| CONVERT_OPT_EXPRESS | NUMBER | 1 if EXPRESS convert option is set, otherwise 0 |
| CONVERT_OPT_NODEADLOCKWAIT | NUMBER | 1 if NODEADLOCKWAIT convert option is set, otherwise 0 |
| CONVERT_OPT_NODEADLOCKBLOCK | NUMBER | 1 if NODEADLOCKBLOCK convert option is set, otherwise 0 |
| WHICH_QUEUE | NUMBER | Which queue the lock is currently located. 0 for NULL queue; 1 for GRANTED queue; 2 for CONVERT queue. |
| STATE | VARCHAR2(64) | State of lock as owner sees it |
| AST_EVENT0 | NUMBER | Last AST event |
| OWNER_NODE | NUMBER | Node identifier |
| BLOCKED | NUMBER | 1 if this lock request is blocked by others, otherwise 0 |
| BLOCKER | NUMBER | 1 if this lock is blocking others, otherwise 0 |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GES_CONVERT_LOCAL

V$GES_CONVERT_LOCAL displays statistics for local GES enqueue operations. This view records average convert times, count information, and timed statistics for global enqueue requests.

| Column | Datatype | Description |
|---|---|---|
| INST_ID | NUMBER | ID of the instance |
| CONVERT_TYPE | VARCHAR2(16) | Conversion type:<br>• NULL -> SS - NULL mode to subshared mode<br>• NULL -> SX - NULL mode to shared exclusive mode<br>• NULL -> S - NULL mode to shared mode<br>• NULL -> SSX - NULL mode to subshared exclusive mode<br>• NULL -> X - NULL mode to exclusive mode<br>• SS -> SX - Subshared mode to shared exclusive mode<br>• SS -> S - Subshared mode to shared mode<br>• SS -> SSX - Subshared mode to subshared exclusive mode<br>• SS -> X - Subshared mode to exclusive mode<br>• SX -> S - Shared exclusive mode to shared mode<br>• SX -> SSX - Shared exclusive mode to subshared exclusive mode<br>• SX -> X - Shared exclusive mode to exclusive mode<br>• S -> SX - Shared mode to shared exclusive mode<br>• S -> SSX - Shared mode to subshared exclusive mode<br>• S -> X - Shared mode to exclusive mode<br>• SSX -> X - Sub-shared exclusive mode to exclusive mode |
| AVERAGE_CONVERT_TIME | NUMBER | Average conversion time for each type of lock operation (in hundredths of a second) |
| CONVERT_COUNT | NUMBER | Number of operations |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GES_CONVERT_REMOTE

V$GES_CONVERT_REMOTE displays values for remote GES enqueue conversions. This view records average convert times, count information, and timed statistics for global enqueue requests.

| Column | Datatype | Description |
|---|---|---|
| INST_ID | NUMBER | ID of the instance |
| CONVERT_TYPE | VARCHAR2(16) | Conversion type:<br>• NULL -> SS - NULL mode to subshared mode<br>• NULL -> SX - NULL mode to shared exclusive mode<br>• NULL -> S - NULL mode to shared mode<br>• NULL -> SSX - NULL mode to subshared exclusive mode<br>• NULL -> X - NULL mode to exclusive mode<br>• SS -> SX - Subshared mode to shared exclusive mode<br>• SS -> S - Subshared mode to shared mode<br>• SS -> SSX - Subshared mode to subshared exclusive mode<br>• SS -> X - Subshared mode to exclusive mode<br>• SX -> S - Shared exclusive mode to shared mode<br>• SX -> SSX - Shared exclusive mode to subshared exclusive mode<br>• SX -> X - Shared exclusive mode to exclusive mode<br>• S -> SX - Shared mode to shared exclusive mode<br>• S -> SSX - Shared mode to subshared exclusive mode<br>• S -> X - Shared mode to exclusive mode<br>• SSX -> X - Sub-shared exclusive mode to exclusive mode |
| AVERAGE_CONVERT_TIME | NUMBER | Average conversion time for each type of lock operation (in hundredths of a second) |
| CONVERT_COUNT | NUMBER | Number of operations |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GES_ENQUEUE

V$GES_ENQUEUE describes all locks currently known to lock manager. This is an Oracle Real Application Clusters view.

**See Also:**

"V$GES_BLOCKING_ENQUEUE" for a description of all such locks that are currently blocking or being blocked

| Column | Datatype | Description |
|---|---|---|
| HANDLE | RAW(4 \| 8) | Lock pointer |
| GRANT_LEVEL | VARCHAR2(9) | Granted level of the lock |
| REQUEST_LEVEL | VARCHAR2(9) | Requested level of the lock |

| Column | Datatype | Description |
|---|---|---|
| RESOURCE_NAME1 | VARCHAR2(30) | Resource name for the lock |
| RESOURCE_NAME2 | VARCHAR2(30) | Resource name for the lock |
| PID | NUMBER | Process identifier which holds the lock |
| TRANSACTION_ID0 | NUMBER | Lower 4 bytes of the transaction identifier to which the lock belongs |
| TRANSACTION_ID1 | NUMBER | Upper 4 bytes of the transaction identifier to which the lock belongs |
| GROUP_ID | NUMBER | Group identifier for the lock |
| OPEN_OPT_DEADLOCK | NUMBER | 1 if DEADLOCK open option is set, otherwise 0 |
| OPEN_OPT_PERSISTENT | NUMBER | 1 if PERSISTENT open option is set, otherwise 0 |
| OPEN_OPT_PROCESS_OWNED | NUMBER | 1 if PROCESS_OWNED open option is set, otherwise 0 |
| OPEN_OPT_NO_XID | NUMBER | 1 if NO_XID open option is set, otherwise 0 |
| CONVERT_OPT_GETVALUE | NUMBER | 1 if GETVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_PUTVALUE | NUMBER | 1 if PUTVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_NOVALUE | NUMBER | 1 if NOVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_DUBVALUE | NUMBER | 1 if DUBVALUE convert option is set, otherwise 0 |
| CONVERT_OPT_NOQUEUE | NUMBER | 1 if NOQUEUE convert option is set, otherwise 0 |
| CONVERT_OPT_EXPRESS | NUMBER | 1 if EXPRESS convert option is set, otherwise 0 |
| CONVERT_OPT_NODEADLOCKWAIT | NUMBER | 1 if NODEADLOCKWAIT convert option is set, otherwise 0 |
| CONVERT_OPT_NODEADLOCKBLOCK | NUMBER | 1 if NODEADLOCKBLOCK convert option is set, otherwise 0 |
| WHICH_QUEUE | NUMBER | Which queue the lock is currently located. 0 for NULL queue; 1 for GRANTED queue; 2 for CONVERT queue. |
| STATE | VARCHAR2(64) | State of the lock as the owner sees it |
| AST_EVENT0 | NUMBER | Last AST event |
| OWNER_NODE | NUMBER | Node identifier |
| BLOCKED | NUMBER | 1 if this lock request is blocked by others, otherwise 0 |
| BLOCKER | NUMBER | 1 if this lock is blocking others, otherwise 0 |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GES_LATCH

V$GES_LATCH is deprecated.

**See Also:**

"V$LATCH" for statistics about GES latch performance

# V$GES_RESOURCE

V$GES_RESOURCE is an Oracle Real Application Clusters view. It displays information of all resources currently known to the lock manager.

| Column | Datatype | Description |
|---|---|---|
| RESP | RAW(4 \| 8) | Resource pointer |
| RESOURCE_NAME | VARCHAR2(30) | Resource name in hexadecimal for the lock |
| ON_CONVERT_Q | NUMBER | 1 if on convert queue, 0 otherwise |
| ON_GRANT_Q | NUMBER | 1 if on granted queue, 0 otherwise |
| PERSISTENT_RES | NUMBER | 1 if it is a persistent resource, 0 otherwise |
| MASTER_NODE | NUMBER | Master node ID |
| NEXT_CVT_LEVEL | VARCHAR2(9) | Next lock level to convert on global convert queue |
| VALUE_BLK_STATE | VARCHAR2(32) | State of the value block |
| VALUE_BLK | VARCHAR2(64) | First 64 bytes of the value block |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GES_STATISTICS

V$GES_STATISTICS displays miscellaneous GES statistics.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC# | NUMBER | Statistic number |
| NAME | VARCHAR2(38) | Name of the statistic |
| VALUE | NUMBER | Value associated with the statistic |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GG_APPLY_COORDINATOR

V$GG_APPLY_COORDINATOR displays information about each GoldenGate apply process coordinator. The coordinator for an apply process gets transactions from the apply process reader and passes them to apply servers. An apply process coordinator is a subcomponent of an apply process used by Oracle GoldenGate Integrated Replicat.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the coordinator's session |
| SERIAL# | NUMBER | Serial number of the coordinator's session |
| STATE | VARCHAR2(21) | State of the coordinator:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• APPLYING - Passing transactions to apply servers<br>• SHUTTING DOWN CLEANLY - Stopping without an error<br>• ABORTING - Stopping because of an apply error |
| APPLY# | NUMBER | Apply process number. An apply process coordinator is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| TOTAL_APPLIED | NUMBER | Total number of transactions applied by the apply process since the apply process was last started |
| TOTAL_WAIT_DEPS | NUMBER | Number of times since the apply process was last started that an apply server waited to apply a logical change record (LCR) in a transaction until another apply server applied a transaction because of a dependency between the transactions |
| TOTAL_WAIT_COMMITS | NUMBER | Number of times since the apply process was last started that an apply server waited to commit a transaction until another apply server committed a transaction to serialize commits |
| TOTAL_ADMIN | NUMBER | Number of administrative jobs issued since the apply process was last started |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_ASSIGNED | NUMBER | Number of transactions assigned to apply servers since the apply process was last started |
| TOTAL_RECEIVED | NUMBER | Total number of transactions received by the coordinator process since the apply process was last started |
| TOTAL_IGNORED | NUMBER | Number of transactions which were received by the coordinator but were ignored because they had been previously applied |
| TOTAL_ROLLBACKS | NUMBER | Number of transactions which were rolled back due to unexpected contention |
| TOTAL_ERRORS | NUMBER | Number of transactions applied by the apply process that resulted in an apply error since the apply process was last started |
| UNASSIGNED_COMPLETE_TXNS | NUMBER | Total number of complete transactions that the coordinator has not assigned to any apply servers |
| LWM_TIME | DATE | Time when the message with the lowest message number was recorded. The creation time of the message with the lowest message number was also recorded at this time. |
| LWM_MESSAGE_NUMBER | NUMBER | Number of the message corresponding to the low watermark. That is, messages with a commit message number less than or equal to this message number have definitely been applied, but some messages with a higher commit message number also may have been applied. |
| LWM_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the message corresponding to the low watermark. For user-enqueued messages, time when the message corresponding to the low watermark was enqueued into the queue at the local database. |
| HWM_TIME | DATE | Time when the message with the highest message number was recorded. The creation time of the message with the highest message number was also recorded at this time. |
| HWM_MESSAGE_NUMBER | NUMBER | Number of the message corresponding to the high watermark. That is, no messages with a commit message number greater than this message number have been applied. |
| HWM_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the message corresponding to the high watermark. For user-enqueued messages, time when the message corresponding to the high watermark was enqueued into the queue at the local database. |
| STARTUP_TIME | DATE | Time when the apply process was last started |
| ELAPSED_SCHEDULE_TIME | NUMBER | Time elapsed (in hundredths of a second) scheduling messages since the apply process was last started |
| ELAPSED_IDLE_TIME | NUMBER | Elapsed idle time |
| LWM_POSITION | RAW(64) | Position of the low-watermark LCR |
| HWM_POSITION | RAW(64) | Position of the high-watermark LCR |

| Column | Datatype | Description |
|---|---|---|
| PROCESSED_MESSAGE_NUMBER | NUMBER | Message number currently processed by the apply coordinator |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| ACTIVE_SERVER_COUNT[1] | NUMBER | Active server count |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

The ELAPSED_SCHEDULE_TIME column is only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$GG_APPLY_READER

V$GG_APPLY_READER displays information about each GoldenGate apply reader. The apply reader is a process which reads (dequeues) messages from the queue, computes message dependencies, and builds transactions. It passes the transactions on to the coordinator in commit order for assignment to the apply servers. An apply reader is a subcomponent of an apply process used by Oracle GoldenGate Integrated Replicat.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the reader's session |
| SERIAL# | NUMBER | Serial number of the reader's session |
| APPLY# | NUMBER | Apply process number. An apply process is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| STATE | VARCHAR2(36) | Shows the state of the apply reader and the hash server. The possible values include:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• DEQUEUE MESSAGES - Dequeuing messages from the queue |

| Column | Datatype | Description |
|---|---|---|
| | | • SCHEDULE MESSAGES - Computing dependencies between messages and assembling messages into transactions<br>• SPILLING - Spilling unapplied messages from memory to hard disk<br>• PAUSED - WAITING FOR DDL TO COMPLETE - Waiting for a data definition language (DDL) logical change record (LCR) to be applied<br>The state of the apply reader is displayed first, followed by the state of the hash server. A semicolon separates the apply reader state from the hash server state. |
| TOTAL_MESSAGES_DEQUEUED | NUMBER | Total number of messages dequeued since the apply process was last started |
| TOTAL_MESSAGES_SPILLED | NUMBER | Number of messages spilled by the reader since the apply process was last started |
| DEQUEUE_TIME | DATE | Time when the last message was received |
| DEQUEUED_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the last message received. For user-enqueued messages, time when the message was enqueued into the queue at the local database. |
| SGA_USED | NUMBER | Amount (in bytes) of SGA memory used by the apply process since it was last started |
| ELAPSED_DEQUEUE_TIME | NUMBER | Time elapsed (in hundredths of a second) dequeuing messages since the apply process was last started |
| ELAPSED_SCHEDULE_TIME | NUMBER | Time elapsed (in hundredths of a second) scheduling messages since the apply process was last started. Scheduling includes computing dependencies between messages and assembling messages into transactions. |
| ELAPSED_SPILL_TIME | NUMBER | Elapsed time (in hundredths of a second) spent spilling messages since the apply process was last started |
| SPILL_LWM_SCN | NUMBER | Spill low-watermark SCN |
| PROXY_SID | NUMBER | When the apply process uses combined capture and apply, the session ID of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is 0. |
| PROXY_SERIAL | NUMBER | When the apply process uses combined capture and apply, the serial number of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is 0. |
| PROXY_SPID | VARCHAR2(12) | When the apply process uses combined capture and apply, the process identification number of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is 0. |
| BYTES_RECEIVED | NUMBER | When the apply process uses combined capture and apply, the number of bytes received by the apply process from the capture |

| Column | Datatype | Description |
|---|---|---|
| | | process since the apply process last started. If the apply process does not use combined capture and apply, then this column is not populated. |
| DEQUEUED_POSITION | RAW(64) | Dequeued position. This column is populated only for an apply process that is functioning as a GoldenGate inbound server. |
| SPILL_LWM_POSITION | RAW(64) | Spill low-watermark position. This column is populated only for an apply process that is functioning as a GoldenGate inbound server. |
| OLDEST_TRANSACTION_ID | VARCHAR2(128) | Oldest transaction ID |
| TOTAL_LCRS_WITH_DEP | NUMBER | Total number of LCRs with row-level dependencies since the apply process last started |
| TOTAL_LCRS_WITH_WMDEP | NUMBER | Total number of LCRs with watermark dependencies since the apply process last started. A watermark dependency occurs when an apply process must wait until the apply process's low watermark reaches a particular threshold. |
| TOTAL_IN_MEMORY_LCRS | NUMBER | Total number of LCRs currently in memory |
| SGA_ALLOCATED | NUMBER | The total amount of shared memory (in bytes) allocated from the GoldenGate pool for the apply process since the apply process last started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The ELAPSED_SCHEDULE_TIME column is only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$GG_APPLY_RECEIVER

V$GG_APPLY_RECEIVER displays information about the message receiver of the Replicat process. The values are reset to zero when the database (or instance in an

Oracle Real Application Clusters (Oracle RAC) environment) restarts and when the Replicat process is stopped.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the apply receiver |
| SERIAL# | NUMBER | Serial number of the apply receiver |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| STARTUP_TIME | DATE | Startup time of the apply process |
| SOURCE_DATABASE_NAME | VARCHAR2(128) | Name of the source database |
| ACKNOWLEDGEMENT | NUMBER | acknowledgment of the messages received by the receiver |
| LAST_RECEIVED_MSG | NUMBER | Last received message |
| TOTAL_MESSAGES_RECEIVED | NUMBER | Total number of messages received |
| TOTAL_AVAILABLE_MESSAGES | NUMBER | Number of available messages |
| STATE | VARCHAR2(46) | State of the apply receiver:<br>• Initializing<br>• Sending unapplied txns<br>• Waiting for message from client<br>• Receiving LCRs<br>• Evaluating rules<br>• Enqueueing LCRS<br>• Waiting for memory<br>• Waiting for apply to read<br>• Waiting for message from Replicat<br>• Waiting for Replicat flush request to complete<br>• Waiting for Replicat commit to complete |
| LAST_RECEIVED_MSG_POSITION | VARCHAR2(64) | Last received message position |
| ACKNOWLEDGEMENT_POSITION | VARCHAR2(64) | acknowledgment position of the messages received by the receiver. Corresponds to ACKNOWLEDGEMENT, except the value is in position rather than SCN. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GG_APPLY_SERVER

V$GG_APPLY_SERVER displays information about each GoldenGate apply server and its activities. An apply server receives messages from the apply coordinator for an

apply process. For each message received, an apply server either applies the message or sends the message to the appropriate apply handler. An apply server is a subcomponent of an apply process used by Oracle GoldenGate Integrated Replicat.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the apply server's session |
| SERIAL# | NUMBER | Serial number of the apply server's session |
| APPLY# | NUMBER | Apply process number. An apply process is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| SERVER_ID | NUMBER | Parallel execution server number of the apply server |
| STATE | VARCHAR2(20) | State of the apply server:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• RECORD LOW-WATERMARK - Performing an administrative job that maintains information about the apply progress, which is used in the ALL_APPLY_PROGRESS and DBA_APPLY_PROGRESS data dictionary views<br>• ADD PARTITION - Performing an administrative job that adds a partition that is used for recording information about in-progress transactions<br>• DROP PARTITION - Performing an administrative job that purges rows that were used to record information about in-progress transactions<br>• EXECUTE TRANSACTION - Applying a transaction<br>• WAIT COMMIT - Waiting to commit a transaction until all other transactions with a lower commit SCN are applied. This state is possible only if the COMMIT_SERIALIZATION apply process parameter is set to a value other than DEPENDENT_TRANSACTIONS and the PARALLELISM apply process parameter is set to a value greater than 1.<br>• WAIT DEPENDENCY - Waiting to apply a logical change record (LCR) in a transaction until another transaction, on which it has a dependency, is applied. This state is possible only if the PARALLELISM apply process parameter is set to a value greater than 1.<br>• ROLLBACK TRANSACTION - Rolling back a transaction<br>• TRANSACTION CLEANUP - Cleaning up an applied transaction, which includes removing LCRs from the apply process's queue<br>• WAIT FOR CLIENT - Waiting for an XStream client application to request more LCRs<br>• WAIT FOR NEXT CHUNK - Waiting for the next set of LCRs for a large transaction |
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction currently being applied |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction currently being applied |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction currently being applied |

| Column | Datatype | Description |
|---|---|---|
| COMMITSCN | NUMBER | Commit system change number (SCN) of the transaction currently being applied |
| DEP_XIDUSN | NUMBER | Transaction ID undo segment number of a transaction on which the transaction being applied by this apply server depends |
| DEP_XIDSLT | NUMBER | Transaction ID slot number of a transaction on which the transaction being applied by this apply server depends |
| DEP_XIDSQN | NUMBER | Transaction ID sequence number of a transaction on which the transaction being applied by this apply server depends |
| DEP_COMMITSCN | NUMBER | Commit system change number (SCN) of the transaction on which this apply server depends |
| MESSAGE_SEQUENCE | NUMBER | Number of the current message being applied by the apply server. This value is reset to 1 at the beginning of each transaction. |
| TOTAL_ASSIGNED | NUMBER | Total number of transactions assigned to the apply server since the apply process was last started |
| TOTAL_ADMIN | NUMBER | Total number of administrative jobs done by the apply server since the apply process was last started. See the STATE information in this view for the types of administrative jobs. |
| TOTAL_ROLLBACKS | NUMBER | Number of transactions assigned to this server which were rolled back |
| TOTAL_MESSAGES_APPLIED | NUMBER | Total number of messages applied by this apply server since the apply process was last started |
| APPLY_TIME | DATE | Time the last message was applied |
| ELAPSED_APPLY_TIME | NUMBER | Time elapsed (in hundredths of a second) applying messages since the apply process was last started |
| COMMIT_POSITION | RAW(64) | Commit position of the transaction. This column is populated only for an apply process that is functioning as a GoldenGate Integrated Replicat. |
| DEP_COMMIT_POSITION | RAW(64) | Commit position of the transaction the slave depends on. This column is populated only for an apply process that is functioning as a GoldenGate inbound server. |
| LAST_APPLY_POSITION | RAW(64) | For inbound servers, the position of the last message applied; for outbound servers, the position of the last message sent to the XStream client application. This column is populated only for an apply process that is functioning as a GoldenGate outbound server or inbound server. |
| TRANSACTION_ID | VARCHAR2(128) | Transaction ID that the slave is applying. This column is populated only for an apply process that is functioning as a GoldenGate inbound server. |
| DEP_TRANSACTION_ID | VARCHAR2(128) | Transaction ID of the transaction the slave depends on. This column is populated only for an apply process that is functioning as a GoldenGate inbound server. |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| TOTAL_LCRS_RETRIED | NUMBER | Total number of LCRs retried by this server |
| LCR_RETRY_ITERATION | NUMBER | Retry iteration for this transaction by this server |
| TOTAL_TXNS_RETRIED | NUMBER | Total transactions retried by this server |
| TXN_RETRY_ITERATION | NUMBER | Retry iteration for this transaction by this server |
| TOTAL_TXNS_RECORDED | NUMBER | Total transactions recorded in error queue by this server |

**Note:**

The ELAPSED_SCHEDULE_TIME column is only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$GLOBAL_BLOCKED_LOCKS

V$GLOBAL_BLOCKED_LOCKS displays global blocked locks.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of lock state object |
| KADDR | RAW(4 \| 8) | Address of lock |
| SID | NUMBER | Identifier of session holding the lock (number) |
| TYPE | VARCHAR2(2) | Resource type (char) |
| ID1 | NUMBER | Resource identifier #1 (number) |
| ID2 | NUMBER | Resource identifier #2 (number) |
| LMODE | NUMBER | Lock mode held (number) |

| Column | Datatype | Description |
|---|---|---|
| REQUEST | NUMBER | Lock mode requested (number) |
| CTIME | NUMBER | Time since current mode was granted |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GLOBAL_TRANSACTION

V$GLOBAL_TRANSACTION displays information on the currently active global transactions.

| Column | Datatype | Description |
|---|---|---|
| FORMATID | NUMBER | Format identifier of the global transaction |
| GLOBALID | RAW(64) | Global transaction identifier of the global transaction |
| BRANCHID | RAW(64) | Branch qualifier of the global transaction |
| BRANCHES | NUMBER | Total number of branches in the global transaction |
| REFCOUNT | NUMBER | Number of siblings for the global transaction (must be the same as branches) |
| PREPARECOUNT | NUMBER | Number of branches of the global transaction that have prepared |
| STATE | VARCHAR2(38) | State of the branch of the global transaction:<br>• ACTIVE<br>• COLLECTING<br>• FINALIZED<br>• FAILED<br>• RECOVERING<br>• UNASSOCIATED<br>• FORGOTTEN<br>• READY FOR RECOVERY<br>• NO-READONLY FAILED<br>• SIBLING INFO WRITTEN<br>• [ORACLE COORDINATED]ACTIVE<br>• [ORACLE COORDINATED]COLLECTING<br>• [ORACLE COORDINATED]FINALIZED<br>• [ORACLE COORDINATED]FAILED<br>• [ORACLE COORDINATED]RECOVERING<br>• [ORACLE COORDINATED]UNASSOCIATED<br>• [ORACLE COORDINATED]FORGOTTEN<br>• [ORACLE COORDINATED]READY FOR RECOVERY<br>• [ORACLE COORDINATED]NO-READONLY FAILED<br>• [MULTINODE]ACTIVE<br>• [MULTINODE]COLLECTING |

| Column | Datatype | Description |
|---|---|---|
| | | • [MULTINODE]FINALIZED<br>• [MULTINODE]FAILED<br>• [MULTINODE]RECOVERING<br>• [MULTINODE]UNASSOCIATED<br>• [MULTINODE]FORGOTTEN<br>• [MULTINODE]READY FOR RECOVERY<br>• [MULTINODE]NO-READONLY FAILED<br>• [MULTINODE]SIBLING INFO WRITTEN<br>• COMBINATION |
| FLAGS | NUMBER | The numeric representation of the state |
| COUPLING | VARCHAR2(15) | Indicates whether the branches are free (FREE), loosely coupled (LOOSELY COUPLED), or tightly coupled (TIGHTLY COUPLED) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$GOLDENGATE_CAPTURE

V$GOLDENGATE_CAPTURE displays information about each capture process that sends LCRs to an Oracle GoldenGate outbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier of the capture process |
| SERIAL# | NUMBER | Session serial number of the capture process session |
| CAPTURE# | NUMBER | Capture process number. A capture process is an Oracle background process prefixed by cp |
| CAPTURE_NAME | VARCHAR2(30) | Name of the capture process |
| LOGMINER_ID | NUMBER | Session ID of the Oracle LogMiner session associated with the capture process |
| STARTUP_TIME | DATE | Time when the capture process was last started |
| STATE | VARCHAR2(589) | State of the capture process and state of the network. The two states are separated by a semicolon. The possible values are:<br>• INITIALIZING - Starting up.<br>• WAITING FOR DICTIONARY REDO - Waiting for redo log files containing the dictionary build related to the first SCN to be added to the capture process session. A capture process cannot begin to scan the redo log files until all of the log files containing the dictionary build have been added.<br>• DICTIONARY INITIALIZATION - Processing a dictionary build.<br>• MINING (PROCESSED SCN = *scn_value*) - Mining a dictionary build at the SCN *scn_value*. |

| Column | Datatype | Description |
|---|---|---|
| | | • LOADING (step *X* of *Y*) - Processing information from a dictionary build and currently at step *X* in a process that involves *Y* steps, where *X* and *Y* are numbers.<br>• CAPTURING CHANGES - Scanning the redo log for changes that satisfy the capture process rule sets.<br>• WAITING FOR REDO - Waiting for new redo log files to be added to the capture process session. The capture process has finished processing all of the redo log files added to its session. This state is possible if there is no activity at a source database. For a downstream capture process, this state is possible if the capture process is waiting for new log files to be added to its session.<br>• EVALUATING RULE - Evaluating a change against a capture process rule set.<br>• CREATING LCR - Converting a change into an LCR.<br>• ENQUEUING MESSAGE - Enqueuing an LCR that satisfies the capture process rule sets into the capture process queue.<br>• PAUSED FOR FLOW CONTROL - Unable to enqueue LCRs either because of low memory or because propagations and outbound servers are consuming messages slower than the capture process is creating them. This state indicates flow control that is used to reduce spilling of captured LCRs when propagation or apply has fallen behind or is unavailable.<br>• WAITING FOR THE BUFFERED QUEUE TO SHRINK - Waiting for the buffered queue to change to a smaller size. The buffered queue shrinks when there is a memory limitation or when an administrator reduces its size.<br>• WAITING FOR *n* SUBSCRIBER(S) INITIALIZING - Waiting for outbound servers that receive LCRs from the capture process to start, where *n* is the number of apply processes.<br>• WAITING FOR TRANSACTION - Waiting for LogMiner to provide more transactions.<br>• WAITING FOR INACTIVE DEQUEUERS - Waiting for the capture process's queue subscribers to start. The capture process stops enqueuing LCRs if there are no active subscribers to the queue.<br>• SUSPENDED FOR AUTO SPLIT/MERGE - Waiting for a merge operation to complete.<br>• SHUTTING DOWN - Stopping.<br>• ABORTING - Aborting. |
| TOTAL_PREFILTER_DISCARDED | NUMBER | Total number of prefiltered messages discarded |
| TOTAL_PREFILTER_KEPT | NUMBER | Total number of prefiltered messages kept |
| TOTAL_PREFILTER_EVALUATIONS | NUMBER | Total number of prefilter evaluations |
| TOTAL_MESSAGES_CAPTURED | NUMBER | Total number of redo entries passed by LogMiner to the capture process for detailed rule evaluation since the capture process last started. A capture process converts a redo entry into a message and performs detailed rule evaluation on the message when capture process prefiltering cannot discard the change. |
| CAPTURE_TIME | DATE | Time when the most recent message was captured |

| Column | Datatype | Description |
|---|---|---|
| CAPTURE_MESSAGE_NUMBER | NUMBER | Number of the most recently captured message |
| CAPTURE_MESSAGE_CREATE_TIME | DATE | Creation time of the most recently captured message |
| TOTAL_MESSAGES_CREATED | NUMBER | Count associated with ELAPSED_LCR_TIME to calculate rate |
| TOTAL_FULL_EVALUATIONS | NUMBER | Count associated with ELAPSED_RULE_TIME to calculate rate |
| TOTAL_MESSAGES_ENQUEUED | NUMBER | Total number of messages enqueued since the capture process was last started |
| ENQUEUE_TIME | DATE | Time when the last message was enqueued |
| ENQUEUE_MESSAGE_NUMBER | NUMBER | Number of the last enqueued message |
| ENQUEUE_MESSAGE_CREATE_TIME | DATE | Creation time of the last enqueued message |
| AVAILABLE_MESSAGE_NUMBER | NUMBER | For local capture, the last redo SCN flushed to the log files. For downstream capture, the last SCN added to LogMiner through the archived redo log files. |
| AVAILABLE_MESSAGE_CREATE_TIME | DATE | For local capture, the time the SCN was written to the log file. For downstream capture, the time the most recent archived redo log file (containing the most recent SCN) was added to LogMiner. |
| ELAPSED_CAPTURE_TIME | NUMBER | Elapsed time (in hundredths of a second) scanning for changes in the redo log since the capture process was last started |
| ELAPSED_RULE_TIME | NUMBER | Elapsed time (in hundredths of a second) evaluating rules since the capture process was last started |
| ELAPSED_ENQUEUE_TIME | NUMBER | Elapsed time (in hundredths of a second) enqueuing messages since the capture process was last started |
| ELAPSED_LCR_TIME | NUMBER | Elapsed time (in hundredths of a second) creating LCRs since the capture process was last started |
| ELAPSED_REDO_WAIT_TIME | NUMBER | Elapsed time (in hundredths of a second) spent by the capture process in the WAITING FOR REDO state |
| ELAPSED_PAUSE_TIME | NUMBER | Elapsed flow control pause time (in hundredths of a second) |
| STATE_CHANGED_TIME | DATE | Time at which the state of the capture process changed |
| SGA_USED | NUMBER | The total amount of shared memory used (in bytes) by the capture process |
| SGA_ALLOCATED | NUMBER | The total amount of shared memory (in bytes) allocated from the Streams pool for the capture process |
| BYTES_OF_REDO_MINED | VARCHAR2(64) | The total amount of redo data mined (in bytes) since the capture process last started |

| Column | Datatype | Description |
|---|---|---|
| SESSION_RESTART_SCN | VARCHAR2(64) | The SCN from which the capture process started mining redo data when it was last started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| SPID | VARCHAR2(12) | Operating system process identifier of the capture process |
| EXTRACT_NAME | VARCHAR2(30) | Name of the extract process |
| SERVER_SID | NUMBER | Session ID of the capture server |
| SERVER_SERIAL# | NUMBER | Session serial number of the capture server |
| SERVER_SPID | VARCHAR2(12) | Operating system process identifier of the capture server |
| TOTAL_MESSAGES_SENT | NUMBER | Total number of LCRs sent by the capture process to the GoldenGate extract process since the last time the extract attached to the capture |
| SEND_TIME | DATE | Time the last LCR was sent by the capture process to the extract process |
| LAST_SENT_MESSAGE_NUMBER | NUMBER | Message number of the last LCR sent by the capture process to the extract process |
| LAST_SENT_MESSAGE_CREATE_TIME | DATE | Creation time at the source database of the last LCR sent by the capture process to the extract process |
| ELAPSED_SEND_TIME | NUMBER | Time elapsed (in hundredths of a second) sending LCRs to the extract process since the last time the extract process attached to the capture process |
| BYTES_SENT | NUMBER | Total number of bytes sent by the capture process to the extract process since the last time the extract process attached to the capture process |

# V$GOLDENGATE_MESSAGE_TRACKING

V$GOLDENGATE_MESSAGE_TRACKING displays information about LCRs tracked through the stream that are processed by Oracle GoldenGate components.

You can track an LCR through a stream using one of the following methods:

- Set the message_tracking_frequency capture process parameter to 1 or another relatively low value.
- Use the DBMS_XSTREAM_ADM.SET_MESSAGE_TRACKING procedure to specify a tracking label that becomes part of each LCR generated by the current session.

  When the actions parameter in the DBMS_XSTREAM_ADM.SET_MESSAGE_TRACKING procedure is set to

DBMS_XSTREAM_ADM.ACTION_MEMORY, information about the LCRs is tracked in memory, and this view is populated with information about the LCRs. Currently, DBMS_XSTREAM_ADM.ACTION_MEMORY is the only valid setting for the actions parameter in the procedure.

| Column | Datatype | Description |
|---|---|---|
| TRACKING_LABEL | VARCHAR2(30) | User-specified tracking label |
| TAG | RAW(30) | First 30 bytes of the tag of the LCR |
| COMPONENT_NAME | VARCHAR2(30) | Name of the component that processed the LCR |
| COMPONENT_TYPE | VARCHAR2(30) | Type of the component that processed the LCR |
| ACTION | VARCHAR2(50) | Action performed on the LCR |
| ACTION_DETAILS | VARCHAR2(100) | Details of the action |
| TIMESTAMP | TIMESTAMP(9) WITH TIME ZONE | Time when the action was performed |
| MESSAGE_CREATION_TIME | DATE | Time when the message was created |
| MESSAGE_NUMBER | NUMBER | SCN of the message |
| TRACKING_ID | RAW(16) | Globally unique OID of the LCR |
| SOURCE_DATABASE_NAME | VARCHAR2(128) | Name of the source database |
| OBJECT_OWNER | VARCHAR2(30) | Owner of the object |
| OBJECT_NAME | VARCHAR2(30) | Name of the object |
| XID | VARCHAR2(128) | Transaction ID |
| COMMAND_TYPE | VARCHAR2(30) | Command type of the LCR |
| MESSAGE_POSITION | RAW(64) | Position of the message |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XSTREAM_ADM.SET_MESSAGE_TRACKING procedure

# V$GOLDENGATE_TABLE_STATS

V$GOLDENGATE_TABLE_STATS displays table statistics for all the tables used by each Oracle GoldenGate apply server.

| Column | Datatype | Description |
|---|---|---|
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| SERVER_ID | NUMBER | Parallel apply server slave ID |
| SOURCE_TABLE_OWNER | VARCHAR2(30) | Source owner of the captured or replicated table |
| SOURCE_TABLE_NAME | VARCHAR2(30) | Source name of the captured or replicated table |
| DESTINATION_TABLE_OWNER | VARCHAR2(30) | Target owner of the captured or replicated table |
| DESTINATION_TABLE_NAME | VARCHAR2(30) | Target name of the captured or replicated table |
| LAST_UPDATE | DATE | Time of the last update of the statistics |
| TOTAL_INSERTS | NUMBER | Number of insert operations on this table processed by this apply server |
| TOTAL_UPDATES | NUMBER | Number of update operations on this table processed by this apply server |
| TOTAL_DELETES | NUMBER | Number of delete operations on this table processed by this apply server |
| INSERT_COLLISIONS | NUMBER | Number of insert collisions on this table encountered by this apply server |
| UPDATE_COLLISIONS | NUMBER | Number of update collisions on this table encountered by this apply server |
| DELETE_COLLISIONS | NUMBER | Number of delete collisions on this table encountered by this apply server |
| REPERROR_RECORDS | NUMBER | Number of change records that resulted in an error that were recorded on this table by this apply server |
| REPERROR_IGNORES | NUMBER | Number of change records that resulted in an error that were ignored on this table by this apply server |
| WAIT_DEPENDENCIES | NUMBER | Number of waits for this table due to dependency |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CDR_INSERT_ROW_EXISTS | NUMBER | Number of conflicts where an insert gets an ORA-00001 error |

| Column | Datatype | Description |
|---|---|---|
| CDR_UPDATE_ROW_EXISTS | NUMBER | Number of conflicts where an update gets an ORA-26786 error |
| CDR_UPDATE_ROW_MISSING | NUMBER | Number of conflicts where an update gets an ORA-26787 error |
| CDR_DELETE_ROW_EXISTS | NUMBER | Number of conflicts where a delete gets an ORA-26786 error |
| CDR_DELETE_ROW_MISSING | NUMBER | Number of conflicts where a delete gets an ORA-26787 error |
| CDR_SUCCESSFUL_RESOLUTIONS | NUMBER | Number of successfully resolved conflicts |
| CDR_FAILED_RESOLUTIONS | NUMBER | Number of conflicts that could not be resolved due to an error during resolution |
| LOB_OPERATIONS[1] | NUMBER | The number of LOB updates (LOB writes, LOB trims, and LOB erases) applied by the inbound server. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$GOLDENGATE_TRANSACTION

V$GOLDENGATE_TRANSACTION displays information about transactions that are being processed by Oracle GoldenGate capture processes, outbound servers, and inbound servers. This view can identify long running transactions and display how many LCRs are being processed in each transaction. This view only contains information about captured LCRs. It does not contain information about user-enqueued LCRs or user messages.

This view only shows information about LCRs that are being processed because they satisfied the rule sets for the component at the time of the query. For capture processes, this view only shows information about changes in transactions that the capture process has converted into LCRs. It does not show information about all the active transactions present in the redo log.

For outbound servers, this view only shows information about LCRs that the outbound server has dequeued. It does not show information about LCRs in the outbound server's queue. For outbound servers, information about a transaction remains in the view until the transaction is sent to the Oracle GoldenGate client application.

For inbound servers, information about a transaction remains in the view until the transaction commits or until the entire transaction is rolled back.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT_NAME | VARCHAR2(30) | Name of the component |
| COMPONENT_TYPE | VARCHAR2(20) | Type of component<br>• CAPTURE - Capture process<br>• APPLY - Apply reader subcomponent in an outbound server or inbound server<br>• PROPAGATION_SENDER - Propagation sender that sends LCRs from a capture process to an outbound server |

| Column | Datatype | Description |
|---|---|---|
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction |
| BATCH_XIDUSN | NUMBER | Transaction ID undo segment number of the batch parent transaction.<br>When using BATCHSQL mode in Oracle GoldenGate Integrated Replicat the BATCH_XID columns identify the parent transaction of the batch the current transaction belongs to. The parent transaction is the first transaction of each batch.<br>This column is populated only if COMPONENT_TYPE is APPLY and BATCHSQL mode is enabled. |
| BATCH_XIDSLT[1] | NUMBER | Transaction ID slot number of the batch parent transaction.<br>When using BATCHSQL mode in Oracle GoldenGate Integrated Replicat the BATCH_XID columns identify the parent transaction of the batch the current transaction belongs to. The parent transaction is the first transaction of each batch.<br>This column is populated only if COMPONENT_TYPE is APPLY and BATCHSQL mode is enabled. |
| BATCH_XIDSQN[1] | NUMBER | Transaction ID sequence number of the batch parent transaction.<br>When using BATCHSQL mode in Oracle GoldenGate Integrated Replicat the BATCH_XID columns identify the parent transaction of the batch the current transaction belongs to. The parent transaction is the first transaction of each batch.<br>This column is populated only if COMPONENT_TYPE is APPLY and BATCHSQL mode is enabled. |
| CUMULATIVE_MESSAGE_COUNT | NUMBER | Number of LCRs processed in the transaction. If a component is restarted while the transaction is being processed, then this column shows the number of LCRs processed in the transaction since the component was started. |
| TOTAL_MESSAGE_COUNT | NUMBER | Total number of LCRs processed in the transaction by an outbound server or inbound server. This column does not pertain to capture processes. |
| FIRST_MESSAGE_TIME | DATE | Time stamp of the first LCR processed in the transaction. If a capture process is restarted while the transaction is being processed, then this column shows the time stamp of the first LCR processed after the capture process was started. |
| FIRST_MESSAGE_NUMBER | NUMBER | SCN of the first message in the transaction. If a capture process is restarted while the transaction is being processed, then this column shows the SCN of the first message processed after the capture process was started. |
| LAST_MESSAGE_TIME | DATE | Time stamp of the last LCR processed in the transaction |
| LAST_MESSAGE_NUMBER | NUMBER | SCN of the most recent message encountered for the transaction |
| FIRST_MESSAGE_POSITION | RAW(64) | Position of the first message seen by an XStream inbound server |

| Column | Datatype | Description |
|---|---|---|
| | | This column is populated only for an apply process that is functioning as an Oracle GoldenGate inbound server. |
| LAST_MESSAGE_POSITION | RAW(64) | Position of the last message seen by an Oracle GoldenGate inbound server<br>This column is populated only for an apply process that is functioning as an Oracle GoldenGate inbound server. |
| TRANSACTION_ID | VARCHAR2(128) | Transaction ID for an Oracle GoldenGate inbound server<br>This column is populated only for an apply process that is functioning as an Oracle GoldenGate inbound server. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$HANG_INFO

V$HANG_INFO displays information about hangs found on the cluster. A hang can be an open wait chain or closed wait chain (cycle or deadlock). A wait chain is a series of sessions that are blocking one another. Each row represents a hang and describes how severe the hang is. This view also includes the victim or final blocker of the hang.

| Column | Datatype | Description |
|---|---|---|
| HANG_ID | NUMBER | A number identifying the hang |
| HANG_CHAIN_SESSIONS | NUMBER | Indicates how many sessions are in the main wait chain of the hang |
| TOTAL_HUNG_SESSIONS | NUMBER | Indicates how many total sessions are affected by the hang, including the main wait chain and any wait chains branching off of the main wait chain |
| HANG_TYPE | VARCHAR2(18) | Is set to Hang if this is an open wait chain or Deadlock if this is a cycle or closed wait chain |
| HANG_CREATE_TIME | VARCHAR2(20) | Date and time that the hang was detected |
| HANG_RESOLVE_TIME | VARCHAR2(20) | Time that the hang may be automatically resolved |
| IGNORED_HANG | VARCHAR2(1) | Y - Hang was ignored and will not be automatically resolved at this time<br>N - The hang has not yet been verified |
| RESOLUTION_ATTEMPTED | VARCHAR2(1) | Y - An attempt was made to resolve the hang<br>N - No attempt was made to resolve the hang |
| GLOBAL_HANG | VARCHAR2(1) | Y - The hang spans multiple nodes in the cluster |

| Column | Datatype | Description |
|---|---|---|
| | | N - All of the sessions in the hang reside on a single instance |
| ESCALATED_HANG | VARCHAR2(1) | Y - An attempt was made to resolve the hang but was unsuccessful, and the hang is becoming more severe. Another resolution attempt will be made.<br>N - The first resolution attempt was successful and it was not necessary to escalate the hang to a higher resolution level. |
| RESOLUTION_STATUS | VARCHAR2(45) | Contains the reason why the hang was resolved or ignored |
| VICTIM_INSTANCE | NUMBER | Instance number of the instance on which the victim or final blocker session resides |
| VICTIM_SESSION_ID | NUMBER | Victim or final blocker's Oracle session ID |
| VICTIM_SERIAL# | NUMBER | Victim or final blocker's Oracle session serial number |
| VICTIM_OSPID | VARCHAR2(24) | Victim or final blocker's operating system process ID |
| FATAL_BACKGROUND | VARCHAR2(1) | Y - Victim or final blocker is a critical background process<br>N - Victim or final blocker is not a critical background process |
| PNAME | VARCHAR2(5) | Name of the victim or final blocker process |
| WAIT_EVENT_TEXT | VARCHAR2(64) | Resource or event for which the victim or final blocker is waiting; set to `not in a wait` if the session is not waiting on a resource |
| VICTIM_QOS_PC_RANK | VARCHAR2(12) | Oracle Database Quality of Service (QoS) Management Performance Class rank of the victim or final blocker of the hang.<br>See *Oracle Database Quality of Service Management User's Guide* for more information about Oracle Database QoS Management. |
| VICTIM_QOS_PC_ITT | NUMBER | Oracle Database QoS Management Performance Class In-Trouble Threshold of the victim or final blocker of the hang |
| VICTIM_QOS_PC_RTT | NUMBER | Oracle Database QoS Management Performance Class Root Termination allowed Threshold of the victim or final blocker of the hang |
| VICTIM_QOS_PC_KEY | NUMBER | Oracle Database QoS Management Performance Class Key of the victim or final blocker of the hang |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$HANG_SESSION_INFO"
- "V$HANG_STATISTICS"

# V$HANG_SESSION_INFO

V$HANG_SESSION_INFO displays information about sessions involved in hangs described by V$HANG_INFO. Each row with the same HANG_ID describes a session that is in the hang wait chain described by the row with the same HANG_ID in V$HANG_INFO. This session is blocked by the victim or final blocker of that hang.

| Column | Datatype | Description |
|---|---|---|
| HANG_ID | NUMBER | A number identifying the hang including this session. This column can be used to join V$HANG_INFO with V$HANG_SESSION_INFO. |
| INSTANCE | NUMBER | Instance number of the instance on which this session resides |
| SID | NUMBER | Oracle session ID of this session |
| SERIAL# | NUMBER | Oracle session serial number of this session |
| OSPID | VARCHAR2(24) | Operating system process ID of this session |
| FATAL_BACKGROUND | VARCHAR2(1) | Y - Session is a critical background process<br>N - Session is not a critical background process |
| ROOT | VARCHAR2(1) | Y - This session is the victim or final blocker of the hang<br>N - This session is not the victim or final blocker of the hang |
| PNAME | VARCHAR2(5) | Name of the victim or final blocker process |
| WAIT_EVENT_TEXT | VARCHAR2(64) | Resource or event for which this session is waiting; set to not in a wait if the session is not waiting on a resource |
| QOS_PC_RANK | VARCHAR2(12) | Session's Oracle Database Quality of Service (QoS) Management Performance Class rank<br>See *Oracle Database Quality of Service Management User's Guide* for more information about Oracle Database QoS Management. |
| QOS_PC_ITT | NUMBER | Session's Oracle Database Quality of Service (QoS) Management Performance Class In-Trouble Threshold |
| QOS_PC_RTT | NUMBER | Session's Oracle Database Quality of Service (QoS) Management Performance Class Root Termination allowed Threshold |
| QOS_PC_KEY | NUMBER | Session's Oracle Database Quality of Service (QoS) Management Performance Class Key |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$HANG_INFO"
- "V$HANG_STATISTICS"

# V$HANG_STATISTICS

V$HANG_STATISTICS displays statistics about hangs found on the cluster.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC# | NUMBER | Statistic number |
| NAME | VARCHAR2(45) | Name of the statistic |
| VALUE | NUMBER | Value associated with the statistic |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$HANG_INFO"
- "V$HANG_SESSION_INFO"

# V$HEAT_MAP_SEGMENT

V$HEAT_MAP_SEGMENT displays real-time segment access information.

| Column | Datatype | Description |
|---|---|---|
| OBJECT_NAME | VARCHAR2(128) | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | Name of the subobject |
| OBJ# | NUMBER | Object number |
| DATAOBJ# | NUMBER | Data object number |
| TS#[1] | NUMBER | ID of the tablespace containing the segment whose heat map is tracked |
| TRACK_TIME | DATE | Timestamp of current activity tracking |
| SEGMENT_WRITE | VARCHAR2(3) | Indicates whether the segment has write access: (YES or NO) |
| SEGMENT_READ | VARCHAR2(3) | Indicates whether the segment has read access: (YES or NO) |

| Column | Datatype | Description |
|---|---|---|
| FULL_SCAN | VARCHAR2(3) | Indicates whether the segment has full table scan: (YES or NO) |
| LOOKUP_SCAN | VARCHAR2(3) | Indicates whether the segment has lookup scan: (YES or NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data<br>The Heat Map feature is not supported in CDBs in Oracle Database 12*c*, so the value in this column can be ignored. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"ALL_HEAT_MAP_SEGMENT"

# V$HM_CHECK

V$HM_CHECK displays information about all the checks registered with Health Monitor. Each check is uniquely identified by a name or an ID.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Health check ID |
| NAME | VARCHAR2(64) | Health check name |
| NAME_NLS | VARCHAR2(1024) | Internationalized names of the health checks |
| CLSID | NUMBER | Class ID to which this check belongs |
| CLS_NAME | VARCHAR2(15) | Class name of the check:<br>• GENERIC<br>• PERSISTENT_DATA |
| FLAGS | NUMBER | Reserved for internal use |
| INTERNAL_CHECK | VARCHAR2(1) | Internal check |
| OFFLINE_CAPABLE | VARCHAR2(1) | Ability to run when database is not open yet (Y or N) |
| DESCRIPTION | VARCHAR2(1024) | Description of what the check does |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HM_CHECK_PARAM

V$HM_CHECK_PARAM displays information about the input parameters of all Health Monitor checks.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Parameter ID |
| NAME | VARCHAR2(64) | Parameter name |
| CHECK_ID | NUMBER | ID of the check to which this parameter belongs |
| TYPE | VARCHAR2(20) | Data type of the input parameter:<br>• DBKH_PARAM_UB4<br>• DBKH_PARAM_UB8<br>• DBKH_PARAM_TEXT<br>• DBKH_PARAM_DATE<br>• DBKH_PARAM_UB4_LIST<br>• DBKH_PARAM_UB8_LIST<br>• DBKH_PARAM_TEXT_LIST<br>• DBKH_PARAM_DATE_LIST |
| DEFAULT_VALUE | VARCHAR2(64) | Default value for this parameter |
| FLAGS | NUMBER | Reserved for internal use |
| DESCRIPTION | VARCHAR2(1024) | Description of the parameter |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HM_FINDING

V$HM_FINDING displays information about all the findings of various Health Monitor runs.

| Column | Datatype | Description |
|---|---|---|
| FINDING_ID | NUMBER | Unique ID to represent the finding |
| RUN_ID | NUMBER | ID of the run that created this finding |
| NAME | VARCHAR2(32) | Name of the finding |

| Column | Datatype | Description |
|---|---|---|
| PARENT_ID | NUMBER | Parent finding ID for this finding |
| CHILD_COUNT | NUMBER | Number of active (open) child findings, if this finding is a parent finding type |
| CLASS_NAME | VARCHAR2(32) | Name of the class to which this finding belongs |
| TIME_DETECTED | TIMESTAMP(6) | Time this finding was detected |
| MODIFIED | TIMESTAMP(6) | Time that this finding was last modified |
| PRIORITY | VARCHAR2(8) | Priority of the finding:<br>• LOW,<br>• HIGH<br>• CRITICAL |
| STATUS | VARCHAR2(12) | Status of the finding:<br>• OPEN<br>• CLOSED |
| TYPE | VARCHAR2(13) | Type of the finding:<br>• INFORMATIONAL<br>• FAILURE |
| DESCRIPTION | VARCHAR2(1024) | Description of the finding |
| DAMAGE_DESCRIPTION | VARCHAR2(1024) | Possible damage description of the finding |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HM_INFO

V$HM_INFO displays information about Health Monitor runs, findings, and recommendations. The information for a run/finding/recommendation is organized as a name, value pair.

If the type of information is RUN, then the data represents the input parameters for that run. If the type of information is FINDING or RECOMMENDATION, then the data represents the information about that particular finding/recommendation.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Unique identifier of the information |
| TYPE | VARCHAR2(14) | Type of the information:<br>• RUN<br>• RUN-RESUME<br>• FINDING |

| Column | Datatype | Description |
|---|---|---|
| | | • RECOMMENDATION |
| NAME | VARCHAR2(32) | Information parameter name |
| VALUE | VARCHAR2(513) | Information parameter value |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HM_RECOMMENDATION

V$HM_RECOMMENDATION displays information about all the recommendations made to various Health Monitor findings.

| Column | Datatype | Description |
|---|---|---|
| RECOMMENDATION_ID | NUMBER | Unique ID to represent the recommendation |
| FDG_ID | NUMBER | Unique ID to represent either the finding ID or the finding set ID for which the recommendation was made |
| RUN_ID | NUMBER | ID of the run that may have generated the recommendation<br>If the ID is 0, then the recommendations were generated by RMAN. |
| NAME | VARCHAR2(32) | Name of the recommendation |
| TYPE | VARCHAR2(7) | Type of the recommendation:<br>• MANUAL<br>• REPAIR |
| RANK | NUMBER | Rank of the recommendation |
| TIME_DETECTED | TIMESTAMP(6) | Time that the recommendation was made |
| EXECUTED | TIMESTAMP(6) | Time that the recommendation (repair) was implemented |
| STATUS | VARCHAR2(7) | Status of the recommendation (repair) execution:<br>• NOT RUN<br>• RUNNING<br>• SUCCESS<br>• FAILED |
| DESCRIPTION | VARCHAR2(1024) | Description of the recommendation |
| REPAIR_SCRIPT | VARCHAR2(512) | Location of the repair script file |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HM_RUN

V$HM_RUN displays information about all Health Monitor checks and their status.

| Column | Datatype | Description |
|---|---|---|
| RUN_ID | NUMBER | Unique ID to represent the run |
| NAME | VARCHAR2(32) | Unique name used to identify the run |
| CHECK_NAME | VARCHAR2(32) | Name of the check |
| RUN_MODE | VARCHAR2(8) | Mode of the run:<br>• MANUAL<br>• REACTIVE<br>• AUTO |
| TIMEOUT | NUMBER | Number of seconds allowed for this run to complete before it is aborted |
| START_TIME | TIMESTAMP(6) | Start time of the run |
| LAST_RESUME_TIME | TIMESTAMP(6) | Last resumed time for the run |
| END_TIME | TIMESTAMP(6) | End time of the run |
| MODIFIED_TIME | TIMESTAMP(6) | Last modified time of the run record |
| STATUS | VARCHAR2(11) | Status of the run:<br>• INITIAL<br>• EXECUTING<br>• INTERRUPTED<br>• TIMEDOUT<br>• CANCELLED<br>• COMPLETED<br>• ERROR |
| SRC_INCIDENT | NUMBER | Source incident ID that activated this run |
| NUM_INCIDENT | NUMBER | Number of incidents created by this run |
| ERROR_NUMBER | NUMBER | Error number if the run failed to complete because of an error |
| PROBLEM_ID | NUMBER | Problem ID of the source incident ID that may have activated this Run |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HS_AGENT

V$HS_AGENT displays the set of HS agents currently running on a given host. There is one row per agent process.

| Column | Datatype | Description |
|---|---|---|
| AGENT_ID | NUMBER | Oracle Net session identifier used for connections to the agent (the identifier used in the LISTENER.ORA file) |
| MACHINE | VARCHAR2(64) | Operating system machine name |
| PROCESS | VARCHAR2(9) | Operating system process identifier of the agent |
| PROGRAM | VARCHAR2(48) | Program name of the agent |
| OSUSER | VARCHAR2(30) | Operating system user |
| STARTTIME | DATE | Starting time |
| AGENT_TYPE | NUMBER | Type of the agent |
| FDS_CLASS_ID | NUMBER | ID of the Foreign Data Store class |
| FDS_INST_ID | NUMBER | Instance name of the Foreign Data Store |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HS_PARAMETER

V$HS_PARAMETER describes the initialization parameters in use by the server and agent.

| Column | Datatype | Description |
|---|---|---|
| HS_SESSION_ID | NUMBER | Unique HS session identifier (maps to the HS_SESSION_ID column of V$HS_SESSION) |
| PARAMETER | VARCHAR2(30) | Name of the parameter |
| VALUE | VARCHAR2(64) | Value of the parameter |
| SOURCE | VARCHAR2(1) | Indicates whether the parameter was defined in the agent (A) or server (S) |

| Column | Datatype | Description |
|---|---|---|
| ENV | VARCHAR2(1) | Indicates whether the parameter was set as an environment variable in the agent (T) or elsewhere (F) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HS_SESSION

V$HS_SESSION describes the current HS session.

| Column | Datatype | Description |
|---|---|---|
| HS_SESSION_ID | NUMBER | Unique HS session identifier |
| AGENT_ID | NUMBER | Oracle Net session identifier used for connections to the agent (maps to the AGENT_ID column of V$HS_AGENT) |
| SID | NUMBER | User session identifier (maps to the SID column of V$SESSION) |
| DB_LINK | VARCHAR2(128) | Server database link name used to access the agent; blank if no database link is used (for example, when using external procedures) |
| DB_LINK_OWNER | NUMBER | Owner of the database link in DB_LINK |
| STARTTIME | DATE | Time the connection was initiated |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$HVMASTER_INFO

V$HVMASTER_INFO describes the current and previous master instances and the number of re-masterings of Global Enqueue Service resources.

| Column | Datatype | Description |
|---|---|---|
| HV_ID | NUMBER | Hash value ID |
| CURRENT_MASTER | NUMBER | Master instance of this hash value ID |
| PREVIOUS_MASTER | NUMBER | Previous master instance of this hash value ID |
| REMASTER_CNT | NUMBER | Number of times this has been remastered |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# 8

# Dynamic Performance (V$) Views: V$IM_COLUMN_LEVEL to V$RULE_SET_AGGREGATE_STATS

This chapter contains the dynamic performance views V$IM_COLUMN_LEVEL to V$RULE_SET_AGGREGATE_STATS.

## V$IM_COLUMN_LEVEL

V$IM_COLUMN_LEVEL presents the selective column compression levels that are defined using the inmemory_memcompress clause of the inmemory_column_clause of the CREATE TABLE statement. This view returns no rows for a table which has no associated selective column compression levels. For each column of a table that is associated with selective column compression levels, this view presents the associated compression level of the column, if any, or the string DEFAULT if no compression level has been defined for the column.

| Column | Datatype | Description |
|---|---|---|
| INST_ID | NUMBER | Instance ID |
| OWNER | VARCHAR2(31) | Username of the table owner |
| OBJ_NUM | NUMBER | Table object number |
| TABLE_NAME | VARCHAR2(31) | Table name |
| SEGMENT_COLUMN_ID | NUMBER | Segment column number |
| COLUMN_NAME | VARCHAR2(31) | Column name |
| INMEMORY_COMPRESSION | VARCHAR2(26) | Column compression level |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "V$IM_SEGMENTS"
- "V$IM_USER_SEGMENTS"
- *Oracle Database Administrator's Guide* for an introduction to the IM column store
- *Oracle Database SQL Language Reference* for more information about the `inmemory_column_clause` of the SQL `CREATE TABLE` statement

# V$IM_SEGMENTS

`V$IM_SEGMENTS` presents information about all the in-memory segments in the database.

Only segments that have an in-memory representation are displayed. If a segment is marked for the In-Memory Column Store (IM column store) but is not populated, no corresponding row for that segment is displayed in this view.

| Column | Datatype | Description |
|---|---|---|
| `OWNER` | `VARCHAR2(128)` | User name of the segment owner |
| `SEGMENT_NAME` | `VARCHAR2(128)` | Name of the segment, if any |
| `PARTITION_NAME` | `VARCHAR2(128)` | Object partition name (set to NULL for non-partitioned objects) |
| `SEGMENT_TYPE` | `VARCHAR2(18)` | Type of segment:<br>• `TABLE`<br>• `TABLE PARTITION`<br>• `TABLE SUBPARTION` |
| `TABLESPACE_NAME` | `VARCHAR2(30)` | Name of the tablespace containing the segment |
| `INMEMORY_SIZE` | `NUMBER` | Size of the in-memory version of the segment, in bytes |
| `BYTES` | `NUMBER` | Total size of the on-disk segment, in bytes |
| `BYTES_NOT_POPULATED` | `NUMBER` | Size of the portion of the on-disk segment that is not populated in memory, in bytes |
| `POPULATE_STATUS` | `VARCHAR2(9)` | Status of segment population:<br>• `STARTED`<br>• `COMPLETED`<br>• `FAILED` |
| `INMEMORY_PRIORITY` | `VARCHAR2(8)` | Indicates the priority for IM column store population:<br>• `LOW`<br>• `MEDIUM` |

| Column | Datatype | Description |
|---|---|---|
| | | • HIGH<br>• CRITICAL<br>• NONE |
| INMEMORY_DISTRIBUTE | VARCHAR2(15) | Indicates how the IM column store is distributed in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| INMEMORY_DUPLICATE | VARCHAR2(13) | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |
| INMEMORY_COMPRESSION | VARCHAR2(17) | Compression level for the IM column store:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ] |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "V$IM_COLUMN_LEVEL"
- "V$IM_USER_SEGMENTS"
- *Oracle Database Administrator's Guide* for an introduction to the IM column store

# V$IM_USER_SEGMENTS

V$IM_USER_SEGMENTS presents information about the in-memory segments for the current user in the database.

Only segments that have an in-memory representation are displayed. If a segment is marked for the In-Memory Column Store (IM column store) but is not populated, no corresponding row for that segment is displayed in this view.

| Column | Datatype | Description |
|---|---|---|
| SEGMENT_NAME | VARCHAR2(128) | Name of the segment, if any |
| PARTITION_NAME | VARCHAR2(128) | Object partition name (set to NULL for non-partitioned objects) |
| SEGMENT_TYPE | VARCHAR2(18) | Type of segment:<br>• TABLE<br>• TABLE PARTITION<br>• TABLE SUBPARTION |
| TABLESPACE_NAME | VARCHAR2(30) | Name of the tablespace containing the segment |
| INMEMORY_SIZE | NUMBER | Size of the in-memory version of the segment, in bytes |
| BYTES | NUMBER | Total size of the on-disk segment, in bytes |
| BYTES_NOT_POPULATED | NUMBER | Size of the portion of the on-disk segment that is not populated in memory, in bytes |
| POPULATE_STATUS | VARCHAR2(9) | Status of segment population:<br>• STARTED<br>• COMPLETED<br>• FAILED |
| INMEMORY_PRIORITY | VARCHAR2(8) | Indicates the priority for IM column store population:<br>• LOW<br>• MEDIUM<br>• HIGH<br>• CRITICAL<br>• NONE |
| INMEMORY_DISTRIBUTE | VARCHAR2(15) | Indicates how the IM column store is distributed in an Oracle Real Application Clusters (Oracle RAC) environment:<br>• AUTO<br>• BY ROWID RANGE<br>• BY PARTITION<br>• BY SUBPARTITION |
| INMEMORY_DUPLICATE | VARCHAR2(13) | Indicates the duplicate setting for the IM column store in an Oracle RAC environment:<br>• NO DUPLICATE<br>• DUPLICATE<br>• DUPLICATE ALL |
| INMEMORY_COMPRESSION | VARCHAR2(17) | Compression level for the IM column store:<br>• NO MEMCOMPRESS<br>• FOR DML<br>• FOR QUERY [ LOW \| HIGH ]<br>• FOR CAPACITY [ LOW \| HIGH ] |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "V$IM_COLUMN_LEVEL"
- "V$IM_SEGMENTS"
- *Oracle Database Administrator's Guide* for an introduction to the IM column store

# V$INDEXED_FIXED_COLUMN

V$INDEXED_FIXED_COLUMN displays the columns in dynamic performance tables that are indexed (X$ tables). The X$ tables can change without notice. Use this view only to write queries against fixed views (V$ views) more efficiently.

| Column | Datatype | Description |
|---|---|---|
| TABLE_NAME | VARCHAR2(30) | Name of the dynamic performance table that is indexed |
| INDEX_NUMBER | NUMBER | Number that distinguishes to which index a column belongs |
| COLUMN_NAME | VARCHAR2(30) | Name of the column that is being indexed |
| COLUMN_POSITION | NUMBER | Position of the column in the index key (this is mostly relevant for multicolumn indexes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$INMEMORY_AREA

V$INMEMORY_AREA contains information on the space allocation inside the In-Memory area.

The In-Memory area is sub-divided into two pools: a 1MB pool used to store the actual column formatted data populated into memory, and a 64K pool used to store metadata about the objects that are populated into the In-Memory Column Store (IM column store). The amount of available memory in each pool is visible in the V $INMEMORY_AREA view. The relative size of the two pools is determined by internal heuristics. The majority of the In-Memory area memory is allocated to the 1MB pool.

| Column | Datatype | Description |
|---|---|---|
| POOL | VARCHAR2(26) | Name of the pools in the In-Memory area |
| ALLOC_BYTES | NUMBER | Total amount of memory allocated to this pool |
| USED_BYTES | NUMBER | Amount of memory currently used in this pool |
| POPULATE_STATUS | VARCHAR2(26) | Shows the status of the IM column store, for example, whether it is currently being populated or if it is done |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "INMEMORY_CLAUSE_DEFAULT"
- "V$IM_COLUMN_LEVEL"
- "V$IM_SEGMENTS"
- "V$IM_USER_SEGMENTS"
- *Oracle Database Administrator's Guide* for an introduction to the IM column store

# V$INSTANCE

V$INSTANCE displays the state of the current instance.

| Column | Datatype | Description |
|---|---|---|
| INSTANCE_NUMBER | NUMBER | Instance number used for instance registration (corresponds to the INSTANCE_NUMBER initialization parameter)<br>**See Also:** "INSTANCE_NUMBER" |

| Column | Datatype | Description |
|---|---|---|
| INSTANCE_NAME | VARCHAR2(16) | Name of the instance |
| HOST_NAME | VARCHAR2(64) | Name of the host machine |
| VERSION | VARCHAR2(17) | Database version |
| STARTUP_TIME | DATE | Time when the instance was started |
| STATUS | VARCHAR2(12) | Status of the instance:<br>• STARTED - After STARTUP NOMOUNT<br>• MOUNTED - After STARTUP MOUNT or ALTER DATABASE CLOSE<br>• OPEN - After STARTUP or ALTER DATABASE OPEN<br>• OPEN MIGRATE - After ALTER DATABASE OPEN { UPGRADE \| DOWNGRADE } |
| PARALLEL | VARCHAR2(3) | Indicates whether the instance is mounted in cluster database mode (YES) or not (NO) |
| THREAD# | NUMBER | Redo thread opened by the instance |
| ARCHIVER | VARCHAR2(7) | Automatic archiving status:<br>• STOPPED<br>• STARTED<br>• FAILED - Archiver failed to archive a log last time but will try again within 5 minutes |
| LOG_SWITCH_WAIT | VARCHAR2(15) | Event that log switching is waiting for:<br>• ARCHIVE LOG<br>• CLEAR LOG<br>• CHECKPOINT<br>• NULL - ALTER SYSTEM SWITCH LOGFILE is hung but there is room in the current online redo log |
| LOGINS | VARCHAR2(10) | Indicates whether the instance is in unrestricted mode, allowing logins by all users (ALLOWED, or in restricted mode, allowing logins by database administrators only (RESTRICTED) |
| SHUTDOWN_PENDING | VARCHAR2(3) | Indicates whether a shutdown is pending (YES) or not (NO) |
| DATABASE_STATUS | VARCHAR2(17) | Status of the database:<br>• ACTIVE<br>• SUSPENDED<br>• INSTANCE RECOVERY |
| INSTANCE_ROLE | VARCHAR2(18) | Indicates whether the instance is an active instance (PRIMARY_INSTANCE) or an inactive secondary instance (SECONDARY_INSTANCE), or UNKNOWN if the instance has been started but not mounted |
| ACTIVE_STATE | VARCHAR2(9) | Quiesce state of the instance:<br>• NORMAL - Database is in a normal state.<br>• QUIESCING - ALTER SYSTEM QUIESCE RESTRICTED has been issued: no new user transactions, queries, or PL/SQL statements are processed in this instance. User transactions, queries, or PL/SQL statements issued before the ALTER SYSTEM QUIESCE RESTRICTED statement are unaffected. |

| Column | Datatype | Description |
|---|---|---|
| | | DBA transactions, queries, or PL/SQL statements are also unaffected.<br>• QUIESCED - ALTER SYSTEM QUIESCE RESTRICTED has been issued: no user transactions, queries, or PL/SQL statements are processed. DBA transactions, queries, or PL/SQL statements are unaffected. User transactions, queries, or PL/ SQL statements issued after the ALTER SYSTEM QUIESCE RESTRICTED statement are not processed.<br>A single ALTER SYSTEM QUIESCE RESTRICTED statement quiesces all instances in an Oracle RAC environment. After this statement has been issued, some instances may enter into a quiesced state before other instances; the system is quiesced when all instances enter the quiesced state. |
| BLOCKED | VARCHAR2(3) | Indicates whether all services are blocked (YES) or not (NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| INSTANCE_MODE | VARCHAR2(11) | Mode of the Oracle Real Application Clusters (Oracle RAC) instance.<br>Possible values:<br>• REGULAR: A regular Oracle RAC instance<br>• READ MOSTLY: An Oracle RAC instance which has very few database writes |
| EDITION | VARCHAR2(7) | The edition of the database.<br>Possible values include:<br>• CORE EE: CORE Enterprise Edition<br>• CORE SE: CORE Standard Edition<br>• EE: Enterprise Edition<br>• PO: Personal Edition<br>• SE: Standard Edition<br>• XE: Express Edition |
| FAMILY | VARCHAR2(80) | For internal use only. |

# V$INSTANCE_CACHE_TRANSFER

V$INSTANCE_CACHE_TRANSFER displays statistics for the cache blocks transferred among instances.

Oracle keeps multiple versions of data buffered in the buffer cache. The current buffer (or block), CURRENT_BLOCK, is the most up-to-date copy, containing all recent modifications. A consistent read buffer (or block), CR_BLOCK, contains the version of the data at a particular time prior to the current buffer. It is read-consistent (that is, all the data shown in that buffer are consistent for the start time of a query).

Therefore, for the same data block there can be multiple copies in the buffer cache: one current copy, and one or more consistent read copies with data consistent as of different snapshot times.

| Column | Datatype | Description |
|---|---|---|
| INSTANCE | NUMBER | Instance from which the blocks are transferred |
| CLASS | VARCHAR2(18) | Class of the cache block |
| LOST | NUMBER | The number of blocks that were sent by a particular instance but that never arrived in this instance |
| LOST_TIME | NUMBER | The time waited for blocks that were sent by a particular instance but that never arrived in this instance |
| CR_BLOCK | NUMBER | CR Block transfers not affected by remote processing delays |
| CR_BLOCK_TIME | NUMBER | Total time waited for CR blocks from a particular instance (includes the other times) |
| CR_2HOP | NUMBER | The count of CR blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CR_2HOP_TIME | NUMBER | The time waited for CR blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CR_3HOP | NUMBER | The count of CR blocks which were received by this instance from a particular instance after a 3-way round-trip |
| CR_3HOP_TIME | NUMBER | The time waited for CR blocks which were received by this instance from a particular instance after a 3-way round-trip |
| CR_BUSY | NUMBER | CR Block transfers affected by remote contention |
| CR_BUSY_TIME | NUMBER | The time waited for CR blocks which were received by this instance from a particular instance and which were delayed by a log flushed on the sending instance |
| CR_CONGESTED | NUMBER | CR Block transfers affected by remote system load |
| CR_CONGESTED_TIME | NUMBER | The time waited for CR blocks which were received by this instance from a particular instance and which were delayed because LMS was busy |
| CURRENT_BLOCK | NUMBER | Current block transfers not affected by remote processing delays |
| CURRENT_BLOCK_TIME | NUMBER | Total time waited for current blocks from a particular instance (includes the other times |
| CURRENT_2HOP | NUMBER | The count of current blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CURRENT_2HOP_TIME | NUMBER | The time waited for current blocks which were received by this instance from a particular instance after a 2-way round-trip |
| CURRENT_3HOP | NUMBER | The count of current blocks which were received by this instance from a particular instance after a 3-way round-trip |
| CURRENT_3HOP_TIME | NUMBER | The time waited for current blocks which were received by this instance from a particular instance after a 3-way round-trip |

| Column | Datatype | Description |
|---|---|---|
| CURRENT_BUSY | NUMBER | Current block transfers affected by remote contention |
| CURRENT_BUSY_TIME | NUMBER | The time waited for current blocks which were received by this instance from a particular instance and which were delayed by a log flushed on the sending instance |
| CURRENT_CONGESTED | NUMBER | Current block transfers affected by remote system load |
| CURRENT_CONGESTED_TIME | NUMBER | The time waited for current blocks which were received by this instance from a particular instance and which were delayed because LMS was busy |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$INSTANCE_PING

V$INSTANCE_PING provides information about measured latency of the interconnect for all instances in an Oracle Real Application Clusters (Oracle RAC) environment.

In an Oracle RAC environment, every few seconds the PING process of each instance checks the response of the interconnect to all instances of the same database.

It sends two messages. One message is 500 bytes in size (referred to in the column descriptions as 500B), and the other is 8 kilobytes in size (referred to in column descriptions as 8K).

For each message sent to each instance, the amount of time it took to get a response back is measured (in microseconds). The view records the latest measurements as well as cumulative data since instance startup.

| Column | Datatype | Description |
|---|---|---|
| INSTANCE | NUMBER | The instance communicated with. In the V$ view for each instance, there is one row for that instance, and in the GV$ view for each instance, there is also one row for that instance, so there are $n^2$ rows for *n* instances. |
| CURRENT_500B | NUMBER | The latest response time for the 500B message (in microseconds) |
| AVERAGE_500B | NUMBER | The average response time for the 500B messages since instance startup (in microseconds). Note that AVERAGE_500B should be the same as WAIT_TIME_500B/COUNT_500B. |
| MAX_500B | NUMBER | The maximal response time for the 500B messages since instance startup (in microseconds) |
| COUNT_500B | NUMBER | The number of measurements for the 500B messages since instance startup |
| WAIT_TIME_500B | NUMBER | The sum of all response times for the 500B messages since instance startup (in microseconds) |

| Column | Datatype | Description |
|---|---|---|
| WAIT_TIME_SQUARED_500B | NUMBER | The sum of the response time squared for 500B messages since instance startup. The unit is in microseconds squared and divided by 1000. |
| CURRENT_8K | NUMBER | The latest response time for the 8K message (in microseconds) |
| AVERAGE_8K | NUMBER | The average response time for the 8K messages since instance startup (in microseconds). Note that AVERAGE_8K should be the same as WAIT_TIME_8K/COUNT_8K. |
| MAX_8K | NUMBER | The maximal response time for the 8K messages since instance startup (in microseconds) |
| COUNT_8K | NUMBER | The number of measurements for the 8K messages since instance startup |
| WAIT_TIME_8K | NUMBER | The sum of all response times for the 8K messages since instance startup (in microseconds) |
| WAIT_TIME_SQUARED_8K | NUMBER | The sum of the response time squared for 8K messages since instance startup. The unit is in microseconds squared and divided by 16. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. For this view, the possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

# V$INSTANCE_RECOVERY

V$INSTANCE_RECOVERY monitors the mechanisms available to users to limit recovery I/O. Those mechanisms are:

- Set the LOG_CHECKPOINT_TIMEOUT initialization parameter
- Set the LOG_CHECKPOINT_INTERVAL initialization parameter
- Set the FAST_START_MTTR_TARGET initialization parameter
- Set the size of the smallest redo log

| Column | Datatype | Description |
|---|---|---|
| RECOVERY_ESTIMATED_IOS | NUMBER | Number of dirty buffers in the buffer cache. In the Standard Edition, this column is always null. |
| ACTUAL_REDO_BLKS | NUMBER | Current actual number of redo blocks required for recovery |
| TARGET_REDO_BLKS | NUMBER | Current target number of redo blocks that must be processed for recovery. This value is the minimum value of the following 3 columns, and identifies which of the 3 user-defined limits determines checkpointing. |
| LOG_FILE_SIZE_REDO_BLKS | NUMBER | Maximum number of redo blocks required to guarantee that a log switch does not occur before the checkpoint completes. |
| LOG_CHKPT_TIMEOUT_REDO_BLKS | NUMBER | Number of redo blocks that need to be processed during recovery to satisfy the LOG_CHECKPOINT_TIMEOUT parameter. |

| Column | Datatype | Description |
|---|---|---|
| | | The value displayed is not meaningful unless that parameter has been set. |
| LOG_CHKPT_INTERVAL_REDO_BLKS | NUMBER | Number of redo blocks that need to be processed during recovery to satisfy the LOG_CHECKPOINT_INTERVAL parameter. The value displayed is not meaningful unless that parameter has been set. |
| FAST_START_IO_TARGET_REDO_BLKS | NUMBER | This column is obsolete and maintained for backward compatibility. The value of this column is always null. |
| TARGET_MTTR | NUMBER | Effective MTTR (mean time to recover) target value in seconds. The TARGET_MTTR value is calculated based on the value of the FAST_START_MTTR_TARGET parameter (the TARGET_MTTR value is used internally), and is usually an approximation of the parameter's value. However, if the FAST_START_MTTR_TARGET parameter value is very small (for example, one second), or very large (for example, 3600 seconds), the calculation will produce a target value dictated by system limitations. In such cases, the TARGET_MTTR value will be the shortest calculated time, or the longest calculated time that recovery is expected to take.<br>If FAST_START_MTTR_TARGET is not specified, the value of this field is the current estimated MTTR. |
| ESTIMATED_MTTR | NUMBER | Current estimated mean time to recover (MTTR) based on the number of dirty buffers and log blocks (0 if FAST_START_MTTR_TARGET is not specified). Basically, this value tells you how long you could expect recovery to take based on the work your system is doing right now. |
| CKPT_BLOCK_WRITES | NUMBER | Number of blocks written by checkpoint writes |
| OPTIMAL_LOGFILE_SIZE | NUMBER | Redo log file size (in megabytes) that is considered optimal based on the current setting of FAST_START_MTTR_TARGET. It is recommended that the user configure all online redo logs to be at least this value. Note that redo log files must be at least 4 megabytes in size; otherwise an error is generated. |
| ESTD_CLUSTER_AVAILABLE_TIME | NUMBER | Estimated time (in seconds) that the cluster would become partially available should this instance fail. This column is only meaningful in an Oracle Real Application Clusters (Oracle RAC) environment. In a non-Oracle RAC environment, the value of this column is null. |
| WRITES_MTTR | NUMBER | Number of writes driven by the FAST_START_MTTR_TARGET initialization parameter |
| WRITES_LOGFILE_SIZE | NUMBER | Number of writes driven by the smallest redo log file size |
| WRITES_LOG_CHECKPOINT_SETTINGS | NUMBER | Number of writes driven by the LOG_CHECKPOINT_INTERVAL or LOG_CHECKPOINT_TIMEOUT initialization parameter |
| WRITES_OTHER_SETTINGS | NUMBER | Number of writes driven by other reasons (such as the deprecated FAST_START_IO_TARGET initialization parameter) |
| WRITES_AUTOTUNE | NUMBER | Number of writes due to auto-tune checkpointing |

| Column | Datatype | Description |
|---|---|---|
| WRITES_FULL_THREAD_CKPT | NUMBER | Number of writes due to full thread checkpoints |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$IO_CALIBRATION_STATUS

V$IO_CALIBRATION_STATUS displays the status of I/O calibration in the instance.

| Column | Datatype | Description |
|---|---|---|
| STATUS | VARCHAR2(13) | Calibration status:<br>• IN PROGRESS - Calibration in Progress (Results from a previous calibration run are displayed, if available)<br>• READY Results are ready and available from an earlier run<br>• NOT AVAILABLE Calibration results are not available |
| CALIBRATION_TIME | TIMESTAMP(3) | End time of the last calibration run |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$IO_OUTLIER

V$IO_OUTLIER contains entries corresponding to I/Os that have taken a long time (more than 500 ms) to complete. Use this view to see if there any occasional delays in serving disk I/O requests by the storage subsystem.

| Column | Datatype | Description |
|---|---|---|
| FUNCTION_NAME | VARCHAR2(18) | I/O function name of the delayed I/O |
| IO_SIZE | NUMBER | Size of the I/O in bytes |
| WAIT_EVENT | VARCHAR2(64) | Wait event name that was used to track the I/O |
| FILE_NAME | VARCHAR2(513) | Name of the file to which the I/O was targeted |
| IO_LATENCY | NUMBER | Time taken to complete the I/O (in milliseconds) |
| DISK1_NAME | VARCHAR2(255) | For Oracle ASM, the name of the first disk to which the I/O was issued |

| Column | Datatype | Description |
|---|---|---|
| DISK1_LATENCY | NUMBER | Latency seen on the first disk (in milliseconds) |
| DISK2_NAME | VARCHAR2(255) | For Oracle ASM, the name of the second disk to which the I/O was issued |
| DISK2_LATENCY | NUMBER | Latency seen on the second disk (in milliseconds) |
| DISK3_NAME | VARCHAR2(255) | For Oracle ASM, the name of the third disk to which the I/O was issued |
| DISK3_LATENCY | NUMBER | Latency seen on the third disk (in milliseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$KERNEL_IO_OUTLIER"
- "V$LGWRIO_OUTLIER"

# V$IOFUNCMETRIC

V$IOFUNCMETRIC displays I/O statistics information by database function for the most recent time interval period.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time for the metric value |
| END_TIME | DATE | End time for the metric value |
| INTSIZE_CSEC | NUMBER | Size of the time period |
| FUNCTION_ID | NUMBER | Function ID |
| FUNCTION_NAME | VARCHAR2(18) | Function name |
| SMALL_READ_MBPS | NUMBER | Single block megabytes read per second |
| SMALL_WRITE_MBPS | NUMBER | Single block megabytes written per second |
| LARGE_READ_MBPS | NUMBER | Multiblock megabytes read per second |
| LARGE_WRITE_MBPS | NUMBER | Multiblock megabytes written per second |
| SMALL_READ_IOPS | NUMBER | Single block read requests per second |
| SMALL_WRITE_IOPS | NUMBER | Single block write requests per second |

| Column | Datatype | Description |
|---|---|---|
| LARGE_READ_IOPS | NUMBER | Multiblock read requests per second |
| LARGE_WRITE_IOPS | NUMBER | Multiblock write requests per second |
| AVG_WAIT_TIME | NUMBER | Average wait time (in milliseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$IOFUNCMETRIC_HISTORY

V$IOFUNCMETRIC_HISTORY displays a recent history of the I/O statistics information by database function for the most recent time interval period.

The columns for V$IOFUNCMETRIC_HISTORY are the same as those for V$IOFUNCMETRIC.

**See Also:**

"V$IOFUNCMETRIC"

## V$IOSTAT_CONSUMER_GROUP

V$IOSTAT_CONSUMER_GROUP displays disk I/O statistics for consumer groups. If the resource manager is enabled, then I/O statistics for all consumer groups that are part of the currently enabled resource plan are captured.

| Column | Datatype | Description |
|---|---|---|
| CONSUMER_GROUP_ID | NUMBER | Consumer group ID |
| SMALL_READ_MEGABYTES | NUMBER | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | Number of multiblock megabytes written |
| SMALL_READ_REQS | NUMBER | Number of single block read requests |
| SMALL_WRITE_REQS | NUMBER | Number of single block write requests |
| LARGE_READ_REQS | NUMBER | Number of multiblock read requests |

| Column | Datatype | Description |
|---|---|---|
| LARGE_WRITE_REQS | NUMBER | Number of multiblock write requests |
| NUMBER_OF_WAITS | NUMBER | Number of I/O waits by consumer group |
| WAIT_TIME | NUMBER | Total wait time (in milliseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$IOSTAT_FILE

V$IOSTAT_FILE displays information about disk I/O statistics of database files (including data files, temp files, and other types of database files). I/O statistics for Data files and Temp files are provided for each file. All other file types (for example, control files, log files, archive logs, and so on) have their statistics consolidated into one entry in the view.

| Column | Datatype | Description |
|---|---|---|
| FILE_NO | NUMBER | File identification number |
| FILETYPE_ID | NUMBER | Type of file (for example, log file, data file, and so on) |
| FILETYPE_NAME | VARCHAR2(28) | Name of the file, in the case of a data file or temp file. For all other files, a corresponding string to be displayed (for example, ARCHIVELOG). |
| SMALL_READ_MEGABYTES | NUMBER | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | Number of multiblock megabytes written |
| SMALL_READ_REQS | NUMBER | Number of single block read requests |
| SMALL_WRITE_REQS | NUMBER | Number of single block write requests |
| SMALL_SYNC_READ_REQS | NUMBER | Number of synchronous single block read requests |
| LARGE_READ_REQS | NUMBER | Number of multiblock read requests |
| LARGE_WRITE_REQS | NUMBER | Number of multiblock write requests |
| SMALL_READ_SERVICETIME | NUMBER | Total service time (in milliseconds) for single block read requests |

| Column | Datatype | Description |
|---|---|---|
| SMALL_WRITE_SERVICETIME | NUMBER | Total service time (in milliseconds) for single block write requests |
| SMALL_SYNC_READ_LATENCY | NUMBER | Latency for single block synchronous reads (in milliseconds) |
| LARGE_READ_SERVICETIME | NUMBER | Total service time (in milliseconds) for multiblock read requests |
| LARGE_WRITE_SERVICETIME | NUMBER | Total service time (in milliseconds) for multiblock write requests |
| ASYNCH_IO | VARCHAR2(9) | Indicates whether asynchronous I/O is available for the file (ASYNC_ON) or not (ASYNC_OFF) |
| ACCESS_METHOD | VARCHAR2(11) | I/O library used to access the file. Possible values include:<br>• OS_LIB - Operating system calls are used to access the file<br>• ODM_LIB - Oracle Disk Manager library is used to access the file<br>• ASM_MANAGED - The file is managed and accessed though ASM<br>• DNFS_LIB - The file is accessed through Direct NFS library |
| RETRIES_ON_ERROR | NUMBER | Number of read retries on error |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$IOSTAT_FUNCTION

V$IOSTAT_FUNCTION displays disk I/O statistics for database functions (such as the LGWR and DBWR).

| Column | Datatype | Description |
|---|---|---|
| FUNCTION_ID | NUMBER | Function ID |
| FUNCTION_NAME | VARCHAR2(18) | Function name:<br>• RMAN<br>• DBWR<br>• LGWR<br>• ARCH<br>• XDB<br>• Streams AQ<br>• Data Pump<br>• Recovery<br>• Buffer Cache Reads<br>• Direct Reads<br>• Direct Writes<br>• Smart Scan |

| Column | Datatype | Description |
|---|---|---|
| | | • Archive Manager<br>• Others |
| SMALL_READ_MEGABYTES | NUMBER | Number of megabytes read via single block read requests |
| SMALL_WRITE_MEGABYTES | NUMBER | Number of megabytes written via single block write requests |
| LARGE_READ_MEGABYTES | NUMBER | Number of megabytes read via multiblock read requests |
| LARGE_WRITE_MEGABYTES | NUMBER | Number of megabytes written via multiblock write requests |
| SMALL_READ_REQS | NUMBER | Number of single block read requests |
| SMALL_WRITE_REQS | NUMBER | Number of single block write requests |
| LARGE_READ_REQS | NUMBER | Number of multiblock read requests |
| LARGE_WRITE_REQS | NUMBER | Number of multiblock write requests |
| NUMBER_OF_WAITS | NUMBER | Number of synchronous I/O waits by functionality |
| WAIT_TIME | NUMBER | Total synchronous I/O wait time (in milliseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$IOSTAT_FUNCTION_DETAIL

V$IOSTAT_FUNCTION_DETAIL displays disk I/O statistics for database functions (such as the LGWR and DBWR), broken down by file type.

| Column | Datatype | Description |
|---|---|---|
| FUNCTION_ID | NUMBER | Function ID |
| FUNCTION_NAME | VARCHAR2(18) | Function name:<br>• RMAN<br>• DBWR<br>• LGWR<br>• ARCH<br>• XDB<br>• Streams AQ<br>• Data Pump<br>• Recovery<br>• Buffer Cache Reads<br>• Direct Reads<br>• Direct Writes |

| Column | Datatype | Description |
|---|---|---|
| | | • Smart Scan<br>• Archive Manager<br>• Others |
| FILETYPE_ID | NUMBER | File type ID |
| FILETYPE_NAME | VARCHAR2(28) | File type name:<br>• Control File<br>• Data File<br>• Log File<br>• Archive Log<br>• Temp File<br>• Data File Backup<br>• Data File Incremental Backup<br>• Archive Log Backup<br>• Data File Copy<br>• Flashback Log<br>• Data Pump Dump File<br>• Other |
| SMALL_READ_MEGABYTES | NUMBER | Number of megabytes read via single block read requests |
| SMALL_WRITE_MEGABYTES | NUMBER | Number of megabytes written via single block write requests |
| LARGE_READ_MEGABYTES | NUMBER | Number of megabytes read via multiblock read requests |
| LARGE_WRITE_MEGABYTES | NUMBER | Number of megabytes written via multiblock write requests |
| SMALL_READ_REQS | NUMBER | Number of single block read requests |
| SMALL_WRITE_REQS | NUMBER | Number of single block write requests |
| LARGE_READ_REQS | NUMBER | Number of multiblock read requests |
| LARGE_WRITE_REQS | NUMBER | Number of multiblock write requests |
| NUMBER_OF_WAITS | NUMBER | Number of synchronous I/O waits by functionality |
| WAIT_TIME | NUMBER | Total synchronous I/O wait time (in milliseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$IOSTAT_NETWORK

V$IOSTAT_NETWORK displays information about network I/O statistics that were caused by accessing files on a remote database instance.

| Column | Datatype | Description |
|---|---|---|
| CLIENT | VARCHAR2(32) | Database client name initiating the network I/O (for example, RMAN or PL/SQL) |
| READS# | NUMBER | Number of read operations issued |
| WRITES# | NUMBER | Number of write operations issued |
| KBYTES_READ | NUMBER | Total number of kilobytes read |
| KBYTES_WRITTEN | NUMBER | Total number of kilobytes written |
| READ_LATENCY | NUMBER | Total read wait time (in milliseconds) |
| WRITE_LATENCY | NUMBER | Total write wait time (in milliseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$JAVA_LIBRARY_CACHE_MEMORY

V$JAVA_LIBRARY_CACHE_MEMORY displays information about memory allocated to library cache memory objects in different namespaces for Java objects. A memory object is an internal grouping of memory for efficient management. A library cache object may consist of one or more memory objects.

| Column | Datatype | Description |
|---|---|---|
| LC_NAMESPACE | VARCHAR2(15) | Library cache namespace |
| LC_INUSE_MEMORY_OBJECTS | NUMBER | Number of library cache memory objects currently in use in the Java pool |
| LC_INUSE_MEMORY_SIZE | NUMBER | Total size of library cache in-use memory objects (in megabytes) |
| LC_FREEABLE_MEMORY_OBJECTS | NUMBER | Number of freeable library cache memory objects in the Java pool |
| LC_FREEABLE_MEMORY_SIZE | NUMBER | Size of library cache freeable memory objects (in megabytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$JAVA_POOL_ADVICE

V$JAVA_POOL_ADVICE displays information about estimated parse time in the Java pool for different pool sizes. The sizes range from 10% of the current Java pool size or the amount of pinned Java library cache memory (whichever is higher) to 200% of the current Java pool size, in equal intervals. The value of the interval depends on the current size of the Java pool.

Parse time saved refers to the amount of time saved by keeping library cache memory objects in the Java pool, as opposed to having to reload these objects.

| Column | Datatype | Description |
|---|---|---|
| JAVA_POOL_SIZE_FOR_ESTIMATE | NUMBER | Java pool size for the estimate (in megabytes) |
| JAVA_POOL_SIZE_FACTOR | NUMBER | Size factor with respect to the current Java pool size |
| ESTD_LC_SIZE | NUMBER | Estimated memory in use by the library cache (in megabytes) |
| ESTD_LC_MEMORY_OBJECTS | NUMBER | Estimated number of library cache memory objects in the Java pool of the specified size |
| ESTD_LC_TIME_SAVED | NUMBER | Estimated elapsed parse time saved (in seconds), owing to library cache memory objects being found in a Java pool of the specified size. This is the time that would have been spent in reloading the required objects in the Java pool had they been aged out due to insufficient amount of available free memory. |
| ESTD_LC_TIME_SAVED_FACTOR | NUMBER | Estimated parse time saved factor with respect to the current Java pool size |
| ESTD_LC_LOAD_TIME | NUMBER | Estimated elapsed time (in seconds) for parsing in a Java pool of the specified size |
| ESTD_LC_LOAD_TIME_FACTOR | NUMBER | Estimated load time factor with respect to the current Java pool size |
| ESTD_LC_MEMORY_OBJECT_HITS | NUMBER | Estimated number of times a library cache memory object was found in a Java pool of the specified size |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$KERNEL_IO_OUTLIER

V$KERNEL_IO_OUTLIER contains entries corresponding to I/Os that have taken a long time (more than 500 ms) to complete. Use this view to see the individual kernel components of I/Os for which there are any occasional delays in serving disk I/O requests by the storage subsystem.

**Note:**

Although this view exists on all platforms in Oracle Database 12*c*, it is only populated on the Solaris platform.

| Column | Datatype | Description |
|---|---|---|
| TIMESTAMP | NUMBER | Number of seconds elapsed since 00:00 UTC, January 1, 1970 |
| IO_SIZE | NUMBER | Size of the I/O, in KB. |
| IO_OFFSET | NUMBER | Offset into the device of the I/O |
| DEVICE_NAME | VARCHAR2(513) | Name of the device to which the I/O was targeted |
| PROCESS_NAME | VARCHAR2(64) | Name of the process that issued the I/O |
| TOTAL_LATENCY | NUMBER | Total time the I/O spent in the kernel (in milliseconds) |
| SETUP_LATENCY | NUMBER | Time spent during initial I/O setup before sending to SCSI target device driver (in milliseconds) |
| QUEUE_TO_HBA_LATENCY | NUMBER | Time spent in the SCSI target device driver before being sent to the Host Bus Adaptor (in milliseconds) |
| TRANSFER_LATENCY | NUMBER | Time spent in the Host Bus Adaptor and physically transferring the I/O to the storage device (in milliseconds) |
| CLEANUP_LATENCY | NUMBER | Time spent freeing resources used by the completed I/O (in milliseconds) |
| PID | NUMBER | Process ID that issued the I/O |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$IO_OUTLIER"
- "V$LGWRIO_OUTLIER"

# V$KEY_VECTOR

V$KEY_VECTOR provides debugging information related to the data structures used by in-memory aggregation for current and recent queries using key vectors.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Session ID |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| TRANSLATE_ID | NUMBER | Translation vector ID |
| SQL_ID | VARCHAR2(13) | SQL ID that uses the translation vector |
| SQL_EXEC_START | DATE | Time when the execution of the SQL started |
| SQL_EXEC_ID | NUMBER | SQL execution identifier |
| PROCESS | VARCHAR2(64) | Operating system client process ID |
| STATE | VARCHAR2(8) | State of the in-memory aggregation operation:<br>• BYPASS<br>Refers to key vectors that are currently in use by queries that have undergone the vector transform.<br>• BYPASSED<br>This is a historical entry for key vectors whose queries have already finished.<br>• FINISHED<br>This is a historical entry for key vectors whose queries have already finished.<br>• RUNNING<br>Refers to key vectors that are currently in use by queries that have undergone the vector transform.<br>One query will probably have multiple key vectors if it has gone through the vector transform.<br>Also, if the query is operating in parallel, there may be multiple entries per PQ slave.<br>The PROCESS column can be joined to V$SESSION to differentiate these cases. |
| TYPE | VARCHAR2(8) | Type of translation vector created:<br>• SIMPLE<br>• INDIRECT<br>• HASH<br>• OFFSET |
| WIDTH | NUMBER | Width (number of bits) of SIMPLE or INDIRECT vector translation array:<br>• 1<br>• 4<br>• 8<br>• 16<br>• 32 |
| KEY_DATA_TYPE | VARCHAR2(13) | The internal join column key data type of either SIMPLE or INDIRECT translation vectors: |

| Column | Datatype | Description |
|---|---|---|
| | | • BINARY<br>• BINARY_FLOAT<br>• NUMBER<br>• PACKED_BINARY<br>• PACKED_DATE<br>• PACKED_NUMBER<br>Also, a value of NULL is displayed for HASH translation vectors with non-numeric data types and for HASH translation vectors with more than one join key column. |
| JOIN_COLUMN_COUNT | NUMBER | Number join columns (can only be 1 for simple or indirect) |
| JOIN_KEY_COUNT | NUMBER | Number of join key values used to build translation vector |
| DUP_JOIN_KEY_COUNT | NUMBER | Indicates the number of key values in the key vector that have more than one parent value |
| MIN_JOIN_KEY | NUMBER | Min join key value in translation vector |
| MAX_JOIN_KEY | NUMBER | Max join key value in translation vector |
| GROUP_KEY_COUNT | NUMBER | Max dense grouping key value. This value is computed while the data that creates the translation vector is processed. |
| FILTERED | NUMBER | Number of rows filtered by translation vector |
| PROBED | NUMBER | Number of rows that probed the translation vector in key vector use row source |
| ACTIVE | NUMBER | Number of active translation vectors used as filters across all slaves |
| DISABLED | NUMBER | Number of translation vector filters across all slaves that were disabled |
| MEMORY_ALLOCATED | NUMBER | Amount of memory allocated for the key vector |
| JOIN_STRUCTURE_SIZE | NUMBER | Amount of memory used out of the allocated space |
| FACT_OWNER | VARCHAR2(128) | The owner of the fact table. Null when the fact table is null. |
| FACT_NAME | VARCHAR2(128) | The table that contains measure data. May be null if more than one fact table is used in the query. |
| DIMENSION_OWNER | VARCHAR2(128) | When a single table is used to construct the key vector, the table owner will appear in this column. If multiple tables are joined to serve as the dimension (more of a snowflake than a star schema shape, for one example), then this column will be null. |
| DIMENSION_NAME | VARCHAR2(128) | The table that contains attribute data and is joined to the fact table. May be null if two or more tables are used (for example, snowflake style dimension tables). |

**Note:**

This view is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

*Oracle Database SQL Tuning Guide* for more information about in-memory aggregation

# V$LATCH

V$LATCH displays aggregate latch statistics for both parent and child latches, grouped by latch name. Individual parent and child latch statistics are broken down in the views V$LATCH_PARENT and V$LATCH_CHILDREN.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of the latch object |
| LATCH# | NUMBER | Latch number |
| LEVEL# | NUMBER | Latch level |
| NAME | VARCHAR2(64) | Latch name |
| HASH | NUMBER | Latch hash |
| GETS | NUMBER | Number of times the latch was requested in willing-to-wait mode |
| MISSES | NUMBER | Number of times the latch was requested in willing-to-wait mode and the requester had to wait |
| SLEEPS | NUMBER | Number of times a willing-to-wait latch request resulted in a session sleeping while waiting for the latch |
| IMMEDIATE_GETS | NUMBER | Number of times a latch was requested in no-wait mode |
| IMMEDIATE_MISSES | NUMBER | Number of times a no-wait latch request did not succeed (that is, missed) |
| WAITERS_WOKEN | NUMBER | This column has been deprecated and is present only for compatibility with previous releases of Oracle. No data is accumulated for this column; it will always have a value of zero. |
| WAITS_HOLDING_LATCH | NUMBER | This column has been deprecated and is present only for compatibility with previous releases of Oracle. No data is accumulated for this column; it will always have a value of zero. |
| SPIN_GETS | NUMBER | Willing-to-wait latch requests which missed the first try but succeeded while spinning |
| SLEEP[1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 \| 10 \| 11] | NUMBER | These columns have been deprecated and are present only for compatibility with previous releases of Oracle. No data is accumulated for these columns; they will always have a value of zero. As a substitute for these columns you can query the appropriate rows of the V$EVENT_HISTOGRAM view where the EVENT column has a value of latch free or latch:%. |
| WAIT_TIME | NUMBER | Elapsed time spent waiting for the latch (in microseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$LATCH_CHILDREN"
- "V$LATCH_PARENT"

# V$LATCH_CHILDREN

V$LATCH_CHILDREN displays statistics about child latches. This view includes all columns of V$LATCH plus the CHILD# column. Note that child latches have the same parent if their LATCH# columns match each other.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of the latch object |
| LATCH# | NUMBER | Latch number of the parent latch |
| CHILD# | NUMBER | Child latch number (unique only to each parent latch) |
| LEVEL# | NUMBER | Latch level |
| NAME | VARCHAR2(64) | Latch name |
| HASH | NUMBER | Latch hash |
| GETS | NUMBER | Number of times the latch was requested in willing-to-wait mode |
| MISSES | NUMBER | Number of times the latch was requested in willing-to-wait mode and the requester had to wait |
| SLEEPS | NUMBER | Number of times a willing-to-wait latch request resulted in a session sleeping while waiting for the latch |
| IMMEDIATE_GETS | NUMBER | Number of times a latch was requested in no-wait mode |
| IMMEDIATE_MISSES | NUMBER | Number of times a no-wait latch request did not succeed (that is, missed) |
| WAITERS_WOKEN | NUMBER | This column has been deprecated and is present only for compatibility with previous releases of Oracle. No data is accumulated for this column; it will always have a value of zero. |
| WAITS_HOLDING_LATCH | NUMBER | This column has been deprecated and is present only for compatibility with previous releases of Oracle. No data is accumulated for this column; it will always have a value of zero. |
| SPIN_GETS | NUMBER | Willing-to-wait latch requests which missed the first try but succeeded while spinning |

| Column | Datatype | Description |
|---|---|---|
| SLEEP[1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 \| 10 \| 11] | NUMBER | These columns have been deprecated and are present only for compatibility with previous releases of Oracle. No data is accumulated for these columns; they will always have a value of zero. As a substitute for these columns you can query the appropriate rows of the V$EVENT_HISTOGRAM view where the EVENT column has a value of latch free or latch:%. |
| WAIT_TIME | NUMBER | Elapsed time spent waiting for the latch (in microseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LATCH"

# V$LATCH_MISSES

V$LATCH_MISSES displays statistics about missed attempts to acquire a latch.

| Column | Datatype | Description |
|---|---|---|
| PARENT_NAME | VARCHAR2(64) | Latch name of a parent latch |
| WHERE | VARCHAR2(80) | This column is obsolete and maintained for backward compatibility. The value of this column is always equal to the value in LOCATION. |
| NWFAIL_COUNT | NUMBER | Number of times that no-wait acquisition of the latch failed |
| SLEEP_COUNT | NUMBER | Number of times that acquisition attempts caused sleeps |
| WTR_SLP_COUNT | NUMBER | Number of times a waiter slept |
| LONGHOLD_COUNT | NUMBER | Number of times someone held a latch for the entire duration of someone else's sleep |
| LOCATION | VARCHAR2(80) | Location that attempted to acquire the latch |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LATCH_PARENT

V$LATCH_PARENT displays statistics about parent latches. The columns for V$LATCH_PARENT are the same as those for V$LATCH.

**See Also:**

"V$LATCH"

# V$LATCHHOLDER

V$LATCHHOLDER displays information about the current latch holders.

| Column | Datatype | Description |
|---|---|---|
| PID | NUMBER | Identifier of the process holding the latch |
| SID | NUMBER | Identifier of the session that owns the latch |
| LADDR | RAW(4 \| 8) | Latch address |
| NAME | VARCHAR2(64) | Name of the latch being held |
| GETS | NUMBER | Number of times that the latch was obtained in either wait mode or no-wait mode |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LATCHNAME

V$LATCHNAME displays information about decoded latch names for the latches shown in V$LATCH. The rows of V$LATCHNAME have a one-to-one correspondence to the rows of V$LATCH.

| Column | Datatype | Description |
|---|---|---|
| LATCH# | NUMBER | Latch number |
| NAME | VARCHAR2(64) | Latch name |
| DISPLAY_NAME | VARCHAR2(64) | A clearer and more descriptive name for the latch that appears in the NAME column. Names that appear in the DISPLAY_NAME column can change across Oracle Database releases, therefore customer scripts should not rely on names that appear in the DISPLAY_NAME column across releases. |
| HASH | NUMBER | Latch hash |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LATCH"

# V$LGWRIO_OUTLIER

V$LGWRIO_OUTLIER contains entries corresponding to Log Writer (LGWR) process I/Os that have taken a long time (more than 500 ms) to complete. Use this view to see if there any occasional delays in serving disk I/O requests by the storage subsystem.

| Column | Datatype | Description |
|---|---|---|
| FUNCTION_NAME | VARCHAR2(18) | I/O function name of the delayed I/O |
| IO_SIZE | NUMBER | Size of the I/O in bytes |
| WAIT_EVENT | VARCHAR2(64) | Wait event name that was used to track the I/O |
| FILE_NAME | VARCHAR2(513) | Name of the file to which the I/O was targeted |
| IO_LATENCY | NUMBER | Time taken to complete the I/O (in milliseconds) |
| DISK1_NAME | VARCHAR2(255) | For Oracle ASM, the name of the first disk to which the I/O was issued |
| DISK1_LATENCY | NUMBER | Latency seen on the first disk (in milliseconds) |
| DISK2_NAME | VARCHAR2(255) | For Oracle ASM, the name of the second disk to which the I/O was issued |
| DISK2_LATENCY | NUMBER | Latency seen on the second disk (in milliseconds) |
| DISK3_NAME | VARCHAR2(255) | For Oracle ASM, the name of the third disk to which the I/O was issued |
| DISK3_LATENCY | NUMBER | Latency seen on the third disk (in milliseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$IO_OUTLIER"
- "V$KERNEL_IO_OUTLIER"

# V$LIBCACHE_LOCKS

V$LIBCACHE_LOCKS displays information about the library cache locks and pins. Locks and pins are distinguished based on the value of the TYPE column.

| Column | Datatype | Description |
|---|---|---|
| TYPE | VARCHAR2(4) | LOCK or PIN |
| ADDR | RAW(4 \| 8) | Address of the lock/pin |
| HOLDING_USER_SESSION | RAW(4 \| 8) | User session holding this lock/pin |
| HOLDING_SESSION | RAW(4 \| 8) | Session holding this lock/pin |
| OBJECT_HANDLE | RAW(4 \| 8) | Handle address for which the lock/pin is acquired |
| LOCK_HELD | RAW(4 \| 8) | If the type is LOCK, then LOCK_HELD represents the pin that is pinning the object.<br>If the type is PIN, then LOCK_HELD represents the lock that is locking the object. |
| REFCOUNT | NUMBER | Reference count for this lock/pin |
| MODE_HELD | NUMBER | Lock/pin mode held:<br>• 0 - No lock/pin held<br>• 1 - Null mode<br>• 2 - Share mode<br>• 3 - Exclusive mode |
| MODE_REQUESTED | NUMBER | Lock/pin mode requested:<br>• 0 - No lock/pin requested<br>• 1 - Null mode<br>• 2 - Share mode<br>• 3 - Exclusive mode |
| SAVEPOINT_NUMBER | NUMBER | Kernel transaction savepoint number at the time the lock/pin was acquired |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LIBRARY_CACHE_MEMORY

V$LIBRARY_CACHE_MEMORY displays information about memory allocated to library cache memory objects in different namespaces. A memory object is an internal grouping of memory for efficient management. A library cache object may consist of one or more memory objects.

| Column | Datatype | Description |
|---|---|---|
| LC_NAMESPACE | VARCHAR2(15) | Library cache namespace |
| LC_INUSE_MEMORY_OBJECTS | NUMBER | Number of library cache memory objects currently in use in the shared pool |
| LC_INUSE_MEMORY_SIZE | NUMBER | Total size of library cache in-use memory objects (in megabytes) |
| LC_FREEABLE_MEMORY_OBJECTS | NUMBER | Number of freeable library cache memory objects in the shared pool |
| LC_FREEABLE_MEMORY_SIZE | NUMBER | Size of library cache freeable memory objects (in megabytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LIBRARYCACHE

V$LIBRARYCACHE contains statistics about library cache performance and activity.

| Column | Datatype | Description |
|---|---|---|
| NAMESPACE | VARCHAR2(64) | Library cache namespace |
| GETS | NUMBER | Number of times a lock was requested for objects of this namespace |
| GETHITS | NUMBER | Number of times an object's handle was found in memory |
| GETHITRATIO | NUMBER | Ratio of GETHITS to GETS |
| PINS | NUMBER | Number of times a PIN was requested for objects of this namespace |
| PINHITS | NUMBER | Number of times all of the metadata pieces of the library object were found in memory |
| PINHITRATIO | NUMBER | Ratio of PINHITS to PINS |
| RELOADS | NUMBER | Any PIN of an object that is not the first PIN performed since the object handle was created, and which requires loading the object from disk |

| Column | Datatype | Description |
|---|---|---|
| INVALIDATIONS | NUMBER | Total number of times objects in this namespace were marked invalid because a dependent object was modified |
| DLM_LOCK_REQUESTS | NUMBER | Number of GET requests lock instance locks |
| DLM_PIN_REQUESTS | NUMBER | Number of PIN requests lock instance locks |
| DLM_PIN_RELEASES | NUMBER | Number of release requests PIN instance locks |
| DLM_INVALIDATION_REQUESTS | NUMBER | Number of GET requests for invalidation instance locks |
| DLM_INVALIDATIONS | NUMBER | Number of invalidation pings received from other instances |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LICENSE

V$LICENSE displays information about license limits.

| Column | Datatype | Description |
|---|---|---|
| SESSIONS_MAX | NUMBER | Maximum number of concurrent user sessions allowed for the instance |
| SESSIONS_WARNING | NUMBER | Warning limit for concurrent user sessions for the instance |
| SESSIONS_CURRENT | NUMBER | Current number of concurrent user sessions |
| SESSIONS_HIGHWATER | NUMBER | Highest number of concurrent user sessions since the instance started |
| USERS_MAX | NUMBER | Maximum number of named users allowed for the database |
| CPU_COUNT_CURRENT | NUMBER | Current number of logical CPUs or processors on the system |
| CPU_CORE_COUNT_CURRENT | NUMBER | Current number of CPU cores on the system (includes subcores of multicore CPUs, as well as single-core CPUs) |
| CPU_SOCKET_COUNT_CURRENT | NUMBER | Current number of CPU sockets on the system (represents an absolute count of CPU chips on the system, regardless of multithreading or multicore architectures) |
| CPU_COUNT_HIGHWATER | NUMBER | Highest number of logical CPUs or processors on the system since the instance started |
| CPU_CORE_COUNT_HIGHWATER | NUMBER | Highest number of CPU cores on the system since the instance started (includes subcores of multicore CPUs, as well as single-core CPUs) |

| Column | Datatype | Description |
|---|---|---|
| CPU_SOCKET_COUNT_HIGHWATER | NUMBER | Highest number of CPU sockets on the system since the instance started (represents an absolute count of CPU chips on the system, regardless of multithreading or multicore architectures) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The availability of the CPU core count and CPU socket count statistics is subject to the operating system platform on which the Oracle Database is running. If a statistic is unavailable, the view will return NULL for the statistic value.

# V$LOADISTAT

V$LOADISTAT contains errors that occurred when updating indexes on a table during a load using the Direct Path API.

| Column | Datatype | Description |
|---|---|---|
| OWNER | VARCHAR2(31) | Schema name |
| TABNAME | VARCHAR2(31) | Table name |
| INDEXNAME | VARCHAR2(31) | Index name |
| SUBNAME | VARCHAR2(31) | Index sub name |
| MESSAGE_NUM | NUMBER | Error message number |
| MESSAGE | VARCHAR2(4000) | Error message |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOADPSTAT

V$LOADPSTAT contains statistics about the number of rows loaded into a partition, or subpartition, during a load using the Direct Path API.

| Column | Datatype | Description |
|---|---|---|
| OWNER | VARCHAR2(31) | Schema name |
| TABNAME | VARCHAR2(31) | Table name |
| PARTNAME | VARCHAR2(31) | Partition name |
| LOADED | NUMBER | Number of rows loaded |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOCK

V$LOCK lists the locks currently held by the Oracle Database and outstanding requests for a lock or latch.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of lock state object |
| KADDR | RAW(4 \| 8) | Address of lock |
| SID | NUMBER | Identifier for session holding or acquiring the lock |
| TYPE | VARCHAR2(2) | Type of user or system lock<br>The locks on the user types are obtained by user applications. Any process that is blocking others is likely to be holding one of these locks. The user type locks are:<br>TM - DML enqueue<br>TX - Transaction enqueue<br>UL - User supplied<br>The system type locks are listed in Table 8-1. Be aware that not all types of locks are documented. To find a complete list of locks for the current release, query the V$LOCK_TYPE data dictionary view, described on "V$LOCK_TYPE". |
| ID1 | NUMBER | Lock identifier #1 (depends on type) |
| ID2 | NUMBER | Lock identifier #2 (depends on type) |
| LMODE | NUMBER | Lock mode in which the session holds the lock:<br>• 0 - none<br>• 1 - null (NULL)<br>• 2 - row-S (SS)<br>• 3 - row-X (SX)<br>• 4 - share (S)<br>• 5 - S/Row-X (SSX)<br>• 6 - exclusive (X) |

| Column | Datatype | Description |
|---|---|---|
| REQUEST | NUMBER | Lock mode in which the process requests the lock:<br>• 0 - none<br>• 1 - null (NULL)<br>• 2 - row-S (SS)<br>• 3 - row-X (SX)<br>• 4 - share (S)<br>• 5 - S/Row-X (SSX)<br>• 6 - exclusive (X) |
| CTIME | NUMBER | Time since current mode was granted |
| BLOCK | NUMBER | Indicates whether the lock in question is blocking other processes. The possible values are:<br>• 0 - The lock is not blocking any other processes<br>• 1 - The lock is blocking other processes<br>• 2 - The lock is not blocking any blocked processes on the local node, but it may or may not be blocking processes on remote nodes. This value is used only in Oracle Real Application Clusters (Oracle RAC) configurations (not in single instance configurations). |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

***Table 8-1 Values for the TYPE Column: System Types***

| System Type | Description | System Type | Description |
|---|---|---|---|
| AE | Edition enqueue | MR | Media recovery |
| AT | Lock held for the ALTER TABLE statement | NA..NZ | Library cache pin instance (A..Z = namespace) |
| BL | Buffer hash table instance | PF | Password File |
| CF | Control file schema global enqueue | PI, PS | Parallel operation |
| CI | Cross-instance function invocation instance | PR | Process startup |
| CU | Cursor bind | QA..QZ | Row cache instance (A..Z = cache) |
| DF | datafile instance | RT | Redo thread global enqueue |
| DL | Direct loader parallel index create | SC | System change number instance |
| DM | Mount/startup db primary/secondary instance | SM | SMON |
| DR | Distributed recovery process | SN | Sequence number instance |

| System Type | Description | System Type | Description |
|---|---|---|---|
| DX | Distributed transaction entry | SQ | Sequence number enqueue |
| FS | File set | SS | Sort segment |
| HW | Space management operations on a specific segment | ST | Space transaction enqueue |
| IN | Instance number | SV | Sequence number value |
| IR | Instance recovery serialization global enqueue | TA | Generic enqueue |
| IS | Instance state | TS | Temporary segment enqueue (ID2=0) |
| IV | Library cache invalidation instance | TS | New block allocation enqueue (ID2=1) |
| JQ | Job queue | TT | Temporary table enqueue |
| KK | Thread kick | UN | User name |
| LA .. LP | Library cache lock instance lock (A..P = namespace) | US | Undo segment DDL |
| MM | Mount definition global enqueue | WL | Being-written redo log instance |

# V$LOCK_ACTIVITY

V$LOCK_ACTIVITY is deprecated. The information that was provided in this view is now provided in the V$INSTANCE_CACHE_TRANSFER and V$SEGMENT_STATISTICS views.

| Column | Datatype | Description |
|---|---|---|
| FROM_VAL | CHAR(4) | Global Cache Resource initial state; always NULL |
| TO_VAL | CHAR(1) | Global Cache Resource initial state; always S |
| ACTION_VAL | CHAR(21) | Description of the conversion; always Lock buffers for read |
| COUNTER | NUMBER | Number of times the lock operation executed |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$INSTANCE_CACHE_TRANSFER"
- "V$SEGMENT_STATISTICS"

# V$LOCK_TYPE

V$LOCK_TYPE describes the type of locks available.

| Column | Datatype | Description |
|---|---|---|
| TYPE | VARCHAR2(64) | A two-letter internal resource identifier |
| NAME | VARCHAR2(64) | Resource type name. This is a short (less than 32 characters) enqueue type name. |
| ID1_TAG | VARCHAR2(64) | Description of the enqueue type. |
| ID2_TAG | VARCHAR2(64) | Further description of the enqueue type. |
| IS_USER | VARCHAR2(3) | User enqueue. These are enqueues that are acquired as a direct result of a SQL statement. Applications may get deadlocks on these enqueues. Such deadlocks are considered application errors. |
| IS_RECYCLE | VARCHAR2(3) | Indicates whether the enqueue requires caching of the associated resource in the DLM's resource cache.<br>Possible values:<br>• NO: Means the enqueue requires caching of the associated resource in the DLM's resource cache<br>• YES: Means the enqueue does not require caching of the associated resource in the DLM's resource cache |
| DESCRIPTION | VARCHAR2(4000) | Explanation of how or for what purpose the enqueue is used. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOCKED_OBJECT

V$LOCKED_OBJECT lists all locks acquired by every transaction on the system. It shows which sessions are holding DML locks (that is, TM-type enqueues) on what objects and in what mode.

| Column | Datatype | Description |
|---|---|---|
| XIDUSN | NUMBER | Undo segment number |
| XIDSLOT | NUMBER | Slot number |

| Column | Datatype | Description |
|---|---|---|
| XIDSQN | NUMBER | Sequence number |
| OBJECT_ID | NUMBER | Object ID being locked |
| SESSION_ID | NUMBER | Session ID |
| ORACLE_USERNAME | VARCHAR2(30) | Oracle user name |
| OS_USER_NAME | VARCHAR2(30) | Operating system user name |
| PROCESS | VARCHAR2(24) | Operating system process ID |
| LOCKED_MODE | NUMBER | Lock mode. The numeric values for this column map to these text values for the lock modes for table locks:<br>• 0 - NONE: lock requested but not yet obtained<br>• 1 - NULL<br>• 2 - ROWS_S (SS): Row Share Lock<br>• 3 - ROW_X (SX): Row Exclusive Table Lock<br>• 4 - SHARE (S): Share Table Lock<br>• 5 - S/ROW-X (SSX): Share Row Exclusive Table Lock<br>• 6 - Exclusive (X): Exclusive Table Lock<br>**See Also:** *Oracle Database Concepts* for more information about lock modes for table locks |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOG

V$LOG displays log file information from the control file.

| Column | Datatype | Description |
|---|---|---|
| GROUP# | NUMBER | Log group number |
| THREAD# | NUMBER | Log thread number |
| SEQUENCE# | NUMBER | Log sequence number |
| BYTES | NUMBER | Size of the log (in bytes) |
| BLOCKSIZE | NUMBER | Block size of the logfile (512 or 4096) |
| MEMBERS | NUMBER | Number of members in the log group |
| ARCHIVED | VARCHAR2(3) | Archive status (YES) or (NO) |
| STATUS | VARCHAR2(16) | Log status:<br>• UNUSED - Online redo log has never been written to. This is the state of a redo log that was just added, or just after a RESETLOGS, when it is not the current redo log. |

| Column | Datatype | Description |
|---|---|---|
| | | • CURRENT - Current redo log. This implies that the redo log is active. The redo log could be open or closed.<br>• ACTIVE - Log is active but is not the current log. It is needed for crash recovery. It may be in use for block recovery. It may or may not be archived.<br>• CLEARING - Log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, the status changes to UNUSED.<br>• CLEARING_CURRENT - Current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch such as an I/O error writing the new log header.<br>• INACTIVE - Log is no longer needed for instance recovery. It may be in use for media recovery. It may or may not be archived. |
| FIRST_CHANGE# | NUMBER | Lowest system change number (SCN) in the log |
| FIRST_TIME | DATE | Time of the first SCN in the log |
| NEXT_CHANGE# | NUMBER | Highest change number (SCN) in the log. When STATUS=CURRENT, NEXT_CHANGE# is set to the highest possible SCN, 281474976710655. |
| NEXT_TIME | DATE | Time of the highest SCN in the log. When STATUS=CURRENT, NEXT_TIME is set to NULL. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOG_HISTORY

V$LOG_HISTORY displays log history information from the control file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Control file record ID |
| STAMP | NUMBER | Control file record stamp |
| THREAD# | NUMBER | Thread number of the archived log |
| SEQUENCE# | NUMBER | Sequence number of the archived log |
| FIRST_CHANGE# | NUMBER | Lowest system change number (SCN) in the log |
| FIRST_TIME | DATE | Time of the first entry (lowest SCN) in the log |
| NEXT_CHANGE# | NUMBER | Highest SCN in the log |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change number of the database when the log was written |

| Column | Datatype | Description |
|---|---|---|
| RESETLOGS_TIME | DATE | Resetlogs time of the database when the log was written |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGFILE

V$LOGFILE contains information about redo log files.

| Column | Datatype | Description |
|---|---|---|
| GROUP# | NUMBER | Redo log group identifier number |
| STATUS | VARCHAR2(7) | Status of the log member:<br>• INVALID - File is inaccessible<br>• STALE - File's contents are incomplete<br>• DELETED - File is no longer used<br>• null - File is in use |
| TYPE | VARCHAR2(7) | Type of the logfile:<br>• ONLINE<br>• STANDBY |
| MEMBER | VARCHAR2(513) | Redo log member name |
| IS_RECOVERY_DEST_FILE | VARCHAR2(3) | Indicates whether the file was created in the fast recovery area (YES) or not (NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGHIST

V$LOGHIST contains log history information from the control file. This view is retained for historical compatibility. Oracle recommends that you use V$LOG_HISTORY instead.

**See Also:**

"V$LOG_HISTORY"

| Column | Datatype | Description |
|---|---|---|
| THREAD# | NUMBER | Log thread number |
| SEQUENCE# | NUMBER | Log sequence number |
| FIRST_CHANGE# | NUMBER | Lowest SCN in the log |
| FIRST_TIME | DATE | Time of first SCN in the log |
| SWITCH_CHANGE# | NUMBER | SCN at which the log switch occurred; one more than highest SCN in the log |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGMNR_CONTENTS

V$LOGMNR_CONTENTS contains log history information. To query this view, you must have the LOGMINING privilege.

When a SELECT statement is executed against the V$LOGMNR_CONTENTS view, the archive redo log files are read sequentially. Translated records from the redo log files are returned as rows in the V$LOGMNR_CONTENTS view. This continues until either the filter criteria specified at startup (EndTime or endScn) are met or the end of the archive log file is reached.

| Column | Datatype | Description |
|---|---|---|
| SCN | NUMBER | System change number (SCN) when the database change was made |
| START_SCN | NUMBER | System change number (SCN) when the transaction that contains this change started; only meaningful if the COMMITTED_DATA_ONLY option was chosen in a DBMS_LOGMNR.START_LOGMNR() invocation, NULL otherwise. This column may also be NULL if the query is run over a time/SCN range that does not contain the start of the transaction. |
| COMMIT_SCN | NUMBER | System change number (SCN) when the transaction committed; only meaningful if the COMMITTED_DATA_ONLY option was chosen in a DBMS_LOGMNR.START_LOGMNR() invocation |
| TIMESTAMP | DATE | Timestamp when the database change was made |
| START_TIMESTAMP | DATE | Timestamp when the transaction that contains this change started; only meaningful if the COMMITTED_DATA_ONLY option was chosen in a DBMS_LOGMNR.START_LOGMNR() invocation, NULL otherwise. This column may also be NULL if the query is run over a time/SCN range that does not contain the start of the transaction. |

| Column | Datatype | Description |
|---|---|---|
| COMMIT_TIMESTAMP | DATE | Timestamp when the transaction committed; only meaningful if the COMMITTED_DATA_ONLY option was chosen in a DBMS_LOGMNR.START_LOGMNR() invocation |
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction that generated the change |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction that generated the change |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction that generated the change |
| XID | RAW(8) | Raw representation of the transaction identifier |
| PXIDUSN | NUMBER | Parent transaction ID undo segment number of a parallel transaction |
| PXIDSLT | NUMBER | Parent transaction ID slot number of a parallel transaction |
| PXIDSQN | NUMBER | Parent transaction ID sequence number of a parallel transaction |
| PXID | RAW(8) | Raw representation of the parent transaction identifier |
| TX_NAME | VARCHAR2(256) | Name of the transaction that made the change; only meaningful if the transaction is a named transaction |
| OPERATION | VARCHAR2(32) | User level SQL operation that made the change:<br>• INSERT - Change was caused by an insert statement<br>• UPDATE - Change was caused by an update statement<br>• DELETE - Change was caused by a delete statement<br>• DDL - Change was caused by a DDL statement<br>• START - Change was caused by the start of a transaction<br>• COMMIT - Change was caused by the commit of a transaction<br>• ROLLBACK - Change was caused by a full rollback of a transaction<br>• LOB_WRITE - Change was caused by an invocation of DBMS_LOB.WRITE<br>• LOB_TRIM - Change was caused by an invocation of DBMS_LOB.TRIM<br>• LOB_ERASE - Change was caused by an invocation of DBMS_LOB.ERASE<br>• SELECT_FOR_UPDATE - Operation was a SELECT FOR UPDATE statement<br>• SEL_LOB_LOCATOR - Operation was a SELECT statement that returned a LOB locator<br>• MISSING_SCN - LogMiner encountered a gap in the redo records. This is most likely because not all redo logs were registered with LogMiner.<br>• INTERNAL - Change was caused by internal operations initiated by the database<br>• XML DOC BEGIN - Beginning of a change to an XMLType column or table<br>• XML DOC WRITE - Data for an XML document<br>• XML DOC END - End of the Data for an XML document |

| Column | Datatype | Description |
|---|---|---|
| | | • UNSUPPORTED - Change was caused by operations not currently supported by LogMiner (for example, changes made to nested tables) |
| OPERATION_CODE | NUMBER | Number of the operation code:<br>• 0 - INTERNAL<br>• 1 - INSERT<br>• 2 - DELETE<br>• 3 - UPDATE<br>• 5 - DDL<br>• 6 - START<br>• 7 - COMMIT<br>• 9 - SELECT_LOB_LOCATOR<br>• 10 - LOB_WRITE<br>• 11 - LOB_TRIM<br>• 25 - SELECT_FOR_UPDATE<br>• 28 - LOB_ERASE<br>• 34 - MISSING_SCN<br>• 68 - XML DOC BEGIN<br>• 70 = XML DOC WRITE<br>• 71 = XML DOC END<br>• 36 - ROLLBACK<br>• 255 - UNSUPPORTED |
| ROLLBACK | NUMBER | 1 = if the redo record was generated because of a partial or a full rollback of the associated transaction<br>0 = otherwise |
| SEG_OWNER | VARCHAR2(32) | Owner of the modified data segment |
| SEG_NAME | VARCHAR2(256) | Name of the modified data segment |
| TABLE_NAME | VARCHAR2(32) | Name of the modified table (in case the redo pertains to a table modification) |
| SEG_TYPE | NUMBER | Type of the modified data segment:<br>• 0 - UNKNOWN<br>• 1 - INDEX<br>• 2 - TABLE<br>• 19 - TABLE PARTITION<br>• 20 - INDEX PARTITION<br>• 34 - TABLE SUBPARTITION<br>• All other values - UNSUPPORTED |
| SEG_TYPE_NAME | VARCHAR2(32) | Segment type name:<br>• UNKNOWN<br>• INDEX<br>• TABLE<br>• TABLE PARTITION<br>• INDEX PARTITION<br>• TABLE SUBPARTITION<br>• UNSUPPORTED |
| TABLE_SPACE | VARCHAR2(32) | Name of the tablespace containing the modified data segment. This column is not populated for rows where the value of the |

| Column | Datatype | Description |
|---|---|---|
| | | OPERATION column is DDL. This is because DDL may operate on more than one tablespace. |
| ROW_ID | VARCHAR2(18) | Row ID of the row modified by the change (only meaningful if the change pertains to a DML). This will be NULL if the redo record is not associated with a DML. |
| USERNAME | VARCHAR2(30) | Name of the user who executed the transaction |
| OS_USERNAME | VARCHAR2(4000) | Name of the operating system user |
| MACHINE_NAME | VARCHAR2(4000) | Machine from which the user connected to the database |
| AUDIT_SESSIONID | NUMBER | Audit session ID associated with the user session making the change |
| SESSION# | NUMBER | Session number of the session that made the change |
| SERIAL# | NUMBER | Serial number of the session that made the change |
| SESSION_INFO | VARCHAR2(4000) | Information about the database session that executed the transaction. Contains process information, machine name from which the user logged in, and so on. A possible SESSION_INFO column may contain the following:<br>• login_username = HR<br>• client_info =<br>• OS_username = jkundu<br>• Machine_name = nirvan<br>• OS_terminal = pts/31<br>• OS_program_name = sqlplus@nirvan (TNS V1-V3) |
| THREAD# | NUMBER | Number of the thread that made the change to the database |
| SEQUENCE# | NUMBER | Sequence number of the SQL statement within the transaction. If you are mining without the COMMITED_DATA_ONLY option set, then this value is 1. |
| RBASQN | NUMBER | Sequence# associated with the Redo Block Address (RBA) of the redo record associated with the change |
| RBABLK | NUMBER | RBA block number within the log file |
| RBABYTE | NUMBER | RBA byte offset within the block |
| UBAFIL | NUMBER | Undo Block Address (UBA) file number identifying the file containing the undo block |
| UBABLK | NUMBER | UBA block number for the undo block |
| UBAREC | NUMBER | UBA record index within the undo block |
| UBASQN | NUMBER | UBA undo block sequence number |
| ABS_FILE# | NUMBER | Data block absolute file number of the block changed by the transaction |

| Column | Datatype | Description |
|---|---|---|
| REL_FILE# | NUMBER | Data block relative file number. The file number is relative to the tablespace of the object. |
| DATA_BLK# | NUMBER | Data block number within the file |
| DATA_OBJ# | NUMBER | Data block object number identifying the object |
| DATA_OBJV# | NUMBER | Version number of the table being modified |
| DATA_OBJD# | NUMBER | Data block data object number identifying the object within the tablespace |
| SQL_REDO | VARCHAR2(4000) | Reconstructed SQL statement that is equivalent to the original SQL statement that made the change. Refer to *Oracle Database Utilities* before executing SQL_REDO to your database.<br>LogMiner does not generate SQL redo for temporary tables. In such a case, this column will contain the string "/* No SQL_REDO for temporary tables */". |
| SQL_UNDO | VARCHAR2(4000) | Reconstructed SQL statement that can be used to undo the effect of the original statement that made the change. DDL statements have no corresponding SQL_UNDO. Refer to *Oracle Database Utilities* before executing SQL_UNDO to your database.<br>LogMiner does not generate SQL undo for temporary tables. In such a case, this column will contain the string "/* No SQL_UNDO for temporary tables */". |
| RS_ID | VARCHAR2(32) | Record set ID. The tuple (RS_ID, SSN) together uniquely identifies a logical row change. This will usually mean one row from V$LOGMNR_CONTENTS, but could be more than one row if a single SQL statement for either the Redo or Undo would be too large to fit within the respective columns SQL_UNDO or SQL_REDO. RS_ID uniquely identifies the redo record that generated the row. |
| SSN | NUMBER | SQL sequence number. Used in conjunction with RS_ID, this uniquely identifies a logical row change, shown as one or more rows from the V$LOGMNR_CONTENTS view. |
| CSF | NUMBER | Continuation SQL flag. Possible values are:<br>• 0 - Indicates SQL_REDO and SQL_UNDO is contained within the same row<br>• 1 - Indicates that either SQL_REDO or SQL_UNDO is greater than 4000 bytes in size and is continued in the next row returned by the view |
| INFO | VARCHAR2(64) | Informational message about the row. For instance, the string "USER DDL" indicates that the DDL statement returned in the SQL_REDO column was the top-level DDL executed by the user and the string "INTERNAL DDL" indicates that the DDL statement returned in the SQL_REDO column was executed internally by the RDBMS. |
| STATUS | NUMBER | A value of 0 indicates that the reconstructed SQL statements as shown in the SQL_REDO and SQL_UNDO columns are valid executable SQL statements. Otherwise, the reconstructed SQL statements are not executable. This may be because no data dictionary was provided to LogMiner for the analysis, or that the |

| Column | Datatype | Description |
|---|---|---|
| | | data dictionary provided did not have the definition of the object being mined.<br>A value of 5 indicates that this row is part of a change to an XMLType column or table and the XML document must be assembled before being applied. |
| REDO_VALUE | NUMBER | Used as input to the DBMS_LOGMNR.MINE_VALUE() and DBMS_LOGMNR.COLUMN_PRESENT() functions |
| UNDO_VALUE | NUMBER | Used as input to the DBMS_LOGMNR.MINE_VALUE() and DBMS_LOGMNR.COLUMN_PRESENT() functions |
| SAFE_RESUME_SCN | NUMBER | Reserved for future use |
| CSCN | NUMBER | This column is deprecated in favor of the COMMIT_SCN column |
| OBJECT_ID | RAW(16) | Object identifier for DMLs to XMLType tables. For changes to non-typed tables, this column is NULL. |
| EDITION_NAME | VARCHAR2(30) | Identifies the edition in which a DDL statement was executed |
| CLIENT_ID | VARCHAR2(64) | Client identifier in the session that performed the operation, if available. |
| SRC_CON_NAME | VARCHAR2(30) | Contains the pluggable database (PDB) name. This information will only be available when mining with a current LogMiner dictionary. |
| SRC_CON_ID | NUMBER | Contains the PDB ID (the PDB_ID column from the DBA_PDBS view). This information will be available only with a current LogMiner dictionary. |
| SRC_CON_UID | NUMBER | Contains the PDB UID (the CON_UID column from the DBA_PDBS view). This information will be available with or without a current LogMiner dictionary. |
| SRC_CON_DBID | NUMBER | Contains the PDB identifier (the DBID column from the DBA_PDBS view). This information will only be available when mining with a current LogMiner dictionary. |
| SRC_CON_GUID | RAW(16) | Contains the GUID associated with the PDB (the GUID column from the DBA_PDBS view). This information will only be available when mining with a current LogMiner dictionary. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_LOGMNR package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_LOB package

# V$LOGMNR_DICTIONARY

V$LOGMNR_DICTIONARY contains log history information.

| Column | Datatype | Description |
|---|---|---|
| DB_NAME | VARCHAR2(9) | Name of the database |
| DB_ID | NUMBER | Database ID |
| DB_CREATED | DATE | Creation date of the source database (corresponds to the CREATED column in the V$DATABASE view) |
| TIMESTAMP | DATE | Date when the dictionary was created |
| RESET_SCN | NUMBER | Reset log SCN when the dictionary was created |
| RESET_SCN_TIME | DATE | Timestamp of the reset log SCN when the dictionary was created |
| DB_VERSION_TIME | DATE | Version time for the source database (corresponds to the VERSION_TIME column in the V$DATABASE view) |
| DB_CHARACTER_SET | VARCHAR2(30) | Character set of the source database |
| DB_VERSION | VARCHAR2(64) | This column is deprecated. |
| DB_STATUS | VARCHAR2(64) | This column is deprecated. |
| DICTIONARY_SCN | NUMBER | Database checkpoint SCN at which the dictionary was created |
| ENABLED_THREAD_MAP | RAW(16) | This column is deprecated. |
| DB_TXN_SCN | NUMBER | SCN at which the dictionary was created |
| FILENAME | VARCHAR2(512) | Dictionary file name |
| INFO | VARCHAR2(32) | Informational/Status message<br>BAD_DATE indicates that the SCN of the dictionary file does not match the SCN range of the log files |
| STATUS | NUMBER | A NULL indicates a valid dictionary file for the list of log files. A non-NULL value indicates further information is contained in the INFO column as a text string. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGMNR_DICTIONARY_LOAD

V$LOGMNR_DICTIONARY_LOAD displays information about LogMiner dictionaries for all active LogMiner sessions on the system. Each query of this view will return one row for each attached LogMiner session. This view will not show valid information for LogMiner adhoc query clients.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | LogMiner session ID |
| LOGMNR_UID | NUMBER | LogMiner dictionary UID |
| ACTION# | NUMBER | Reserved for internal use |
| OPCODE | NUMBER | Reserved for internal use |
| COMMAND | VARCHAR2(161) | Current command being executed |
| CURRENT_STATE | VARCHAR2(32) | Summary information if LOADED=ACTIVE |
| COMPLETED_ACTIONS | NUMBER | The steps completed so far |
| TOTAL_ACTIONS | NUMBER | Total steps to complete |
| LOADED | VARCHAR2(7) | Status of the dictionary load:<br>• YES - dictionary has been loaded<br>• NO - dictionary has not been loaded<br>• ACTIVE - dictionary is in the process of being loaded |
| PERCENT_DONE | NUMBER | Percentage of the dictionary that has been loaded |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGMNR_LATCH

V$LOGMNR_LATCH can be joined with the V$LATCH and the V$LATCH_CHILDREN views to obtain statistics about different latches used by active LogMiner persistent sessions. A persistent LogMiner session is created either by starting Data Guard SQL Apply on a logical standby database for the first time or by creating a Streams capture.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Unique identifier of the LogMiner persistent session |
| NAME | VARCHAR2(32) | Name of the Latch: |

| Column | Datatype | Description |
|---|---|---|
| | | • `LogMiner internal state` - Identifies the latch that protects communications between the LogMiner `READER`, `PREPARER`, and `BUILDER` processes<br>• `LogMiner memory allocation` - Identifies the latch that protects all memory allocation and deallocation inside LogMiner<br>• `LogMiner transaction list` - Identifies the latch that protects interaction between the LogMiner layer and its clients (Data Guard SQL Apply on a logical standby database or Oracle Streams) |
| `CHILD_ADDR` | `RAW(4 \| 8)` | Address of the child latch object. This column matches the corresponding `ADDR` column in the `V$LATCH_CHLDREN` view |
| `STATE` | `VARCHAR2(6)` | State of the Latch:<br>• `UNINIT` - The latch structure is uninitialized; that is, it is not currently assigned to any LogMiner persistent session<br>• `READY` - The latch structure is being used by a persistent LogMiner session |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$LATCH"
- "V$LATCH_CHILDREN"

# V$LOGMNR_LOGS

`V$LOGMNR_LOGS` contains log information.

| Column | Datatype | Description |
|---|---|---|
| `LOG_ID` | `NUMBER` | This column is deprecated. |
| `FILENAME` | `VARCHAR2(512)` | Name of the log file |
| `LOW_TIME` | `DATE` | Oldest date of any records in the file |
| `HIGH_TIME` | `DATE` | Most recent date of any records in the file |
| `DB_ID` | `NUMBER` | Database ID |
| `DB_NAME` | `VARCHAR2(8)` | Name of the database |
| `RESET_SCN` | `NUMBER` | Resetlogs SCN of the database incarnation that generated the log file |

| Column | Datatype | Description |
|---|---|---|
| RESET_SCN_TIME | DATE | Resetlogs timestamp of the database incarnation that generated the log file |
| COMPATIBLE | VARCHAR2(17) | The setting of the database COMPATIBLE initialization parameter at the time the log file was generated |
| THREAD_ID | NUMBER | Thread number |
| THREAD_SQN | NUMBER | Thread sequence number |
| LOW_SCN | NUMBER | SCN allocated when log switched into |
| NEXT_SCN | NUMBER | SCN after this log. Low SCN of the next log. |
| DICTIONARY_BEGIN | VARCHAR2(3) | Indicates whether dictionary dumped to redo logs starts in this redo log (YES) or not (NO) |
| DICTIONARY_END | VARCHAR2(3) | Indicates whether dictionary dumped to redo logs ends in this redo log (YES) or not (NO) |
| TYPE | VARCHAR2(7) | Redo log file type:<br>• ARCHIVED<br>• ONLINE |
| BLOCKSIZE | NUMBER | Database block size |
| FILESIZE | NUMBER | Size of the redo file (in bytes) |
| INFO | VARCHAR2(32) | Informational message. A value of MISSING_LOGFILE will be assigned to a row entry where a needed log file is missing from the list of log files. |
| STATUS | NUMBER | Status of the redo log file:<br>• 0 - Will be read<br>• 1 - First to be read<br>• 2 - Not needed<br>• 4 - Missing log file |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGMNR_PARAMETERS

V$LOGMNR_PARAMETERS contains log information.

| Column | Datatype | Description |
|---|---|---|
| START_DATE | DATE | Date to start the search |
| REQUIRED_START_DATE | DATE | Required date to start the search if DDL tracking is enabled |

| Column | Datatype | Description |
|---|---|---|
| END_DATE | DATE | Date to end the search |
| START_SCN | NUMBER | System change number to start the search |
| REQUIRED_START_SCN | NUMBER | Required system change number to start the search if DDL tracking is enabled |
| END_SCN | NUMBER | System change number to end the search |
| OPTIONS | NUMBER | Options specified for the current LogMiner session |
| INFO | VARCHAR2(32) | This column is always null. |
| STATUS | NUMBER | This column is always 0. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGMNR_PROCESS

V$LOGMNR_PROCESS identifies all processes attached to an active LogMiner persistent session. (A persistent LogMiner session is created either by starting Data Guard SQL Apply on a logical standby database for the first time or by creating Streams capture.) This view can be joined with either the V$SESSION view or the V$PROCESS view to gather process-specific information.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Unique identifier for the LogMiner persistent session |
| PID | NUMBER | Oracle process identifier for the SQL Apply or Streams capture process (same as the V$PROCESS.PID) |
| SPID | VARCHAR2(24) | Operating system process identifier (same as the V$PROCESS.SPID) |
| ROLE | VARCHAR2(32) | Identifies the role of the active LogMiner process: READER, PREPARER, BUILDER, COORDINATOR, or APPLY SERVER |
| USERNAME | VARCHAR2(15) | Operating system process user name that is connected to the database |
| SID | NUMBER | Session identifier for the V$SESSION.SID process |
| SERIAL# | NUMBER | Session serial number associated with the V$SESSION.SERIAL process |
| LATCHWAIT | VARCHAR2(16) | Address of the latch the process is waiting for; NULL if none |
| LATCHSPIN | VARCHAR2(16) | This column is obsolete |
| WORK_MICROSEC | VARCHAR2(21) | Microseconds spent by the process doing useful work |

| Column | Datatype | Description |
|---|---|---|
| OVERHEAD_MICROSEC | VARCHAR2(21) | Microseconds spent by the process doing overhead tasks or simply waiting/idling |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SESSION"
- "V$PROCESS"

# V$LOGMNR_SESSION

V$LOGMNR_SESSION displays information about active LogMiner persistent sessions. (A LogMiner persistent session is created either by starting Data Guard SQL Apply on a logical standby database for the first time or by creating Streams Capture.) Transient LogMiner sessions (those created as a result of querying the V$LOGMNR_CONTENTS view) do not show up in the V$LOGMNR_SESSION view. The statistics shown in this view correspond to snapshots of the system and are not cumulative in nature.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Unique identifier for the LogMiner persistent session |
| SESSION_NAME | VARCHAR2(32) | Unique session name |
| SESSION_STATE | VARCHAR2(9) | Current state of the LogMiner persistent session:<br>• READY - Client has created the LogMiner persistent session and added the relevant archived redo log files, but has not loaded the initial LogMiner dictionary<br>• STARTED - The LogMiner dictionary has been loaded<br>• ACTIVE - The LogMiner persistent session is mining the redo stream<br>• DETACHED - The LogMiner persistent session is not currently active, and it is in the process of becoming inactive<br>• DISCARDED - Client is about to destroy the LogMiner persistent session |
| DB_NAME | VARCHAR2(128) | Global database name for the source database |
| DB_ID | NUMBER | Database identifier of the source database |
| RESET_SCN | NUMBER | System change number (SCN) when the session started |
| RESET_TIMESTAMP | NUMBER | Time of the RESETLOGs when the LogMiner persistent session started |

| Column | Datatype | Description |
|---|---|---|
| RESET_TIME | DATE | The time when the session started |
| NUM_PROCESS | NUMBER | Number of processes allocated to this session |
| CHUNK_SIZE | NUMBER | Amount of memory allocated for this chunk |
| START_SCN | NUMBER | System change number (SCN) at start of the session |
| END_SCN | NUMBER | System change number (SCN) at end of the session |
| SPILL_SCN | NUMBER | In the event of a restart, redo records with an SCN lower than this will not be read from the archived redo log files |
| PROCESSED_SCN | NUMBER | The BUILDER process has successfully mined redo records up to this SCN |
| PROCESSED_TIME[1] | DATE | The BUILDER process has successfully mined redo records up to this time |
| PREPARED_SCN | NUMBER | The PREPARER processes have successfully transformed all redo records below this SCN into logical change records (LCRs). However the LCRs may not have been grouped into transactions or merged in case they pertain to chained rows or LOB updates. |
| READ_SCN | NUMBER | The READER process has read all redo records below this SCN |
| LOW_MARK_SCN | NUMBER | LogMiner has delivered all transactions that committed below this SCN to the client |
| CONSUMED_SCN | NUMBER | Client has consumed and released all transactions that committed below this SCN. |
| MAX_MEMORY_SIZE | NUMBER | Maximum amount of shared memory (in bytes) that LogMiner is allowed to consume |
| USED_MEMORY_SIZE | NUMBER | Amount of shared memory (in bytes) actually consumed by LogMiner |
| BUILDER_WORK_SIZE | NUMBER | Amount of redo (in bytes) in the current work unit being processed by the BUILDER process. |
| PREPARED_WORK_SIZE | NUMBER | Amount of redo (in bytes) that has been prepared by LogMiner |
| AVAILABLE_WORK_SIZE | NUMBER | Amount of redo records (in bytes) that are ready, but are yet to be consumed by the client |
| AVAILABLE_TXN | NUMBER | Number of transaction chunks ready for consumption |
| AVAILABLE_COMMITTED_TXN | NUMBER | Number of committed transactions ready for consumption. This is less than, or equal to, AVAILABLE_TXN. |
| DELIVERED_TXN | NUMBER | Number of transaction chunks that the client currently has in its possession |
| DELIVERED_COMMITTED_TXN | NUMBER | Number of committed transaction chunks that the client is currently working on. This is less than, or equal to, the value of the DELIVERED_TXN column. |
| PINNED_TXN | NUMBER | Number of transactions pinned (the client is actively working on) |

| Column | Datatype | Description |
|---|---|---|
| PINNED_COMMITTED_TXN | NUMBER | Number of committed transactions pinned (the client is actively working on) |
| CHECKPOINT_INTERVAL | NUMBER | Checkpoint interval |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

"V$LOGMNR_CONTENTS"

# V$LOGMNR_STATS

V$LOGMNR_STATS displays the activity currently being performed by the active LogMiner persistent sessions.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Unique identifier for the LogMiner persistent sessions |
| NAME | VARCHAR2(64) | Name of the LogMiner statistic, state, or status, including:<br>• Bytes of Redo Processed - Cumulative count of bytes processed by SQL Apply<br>• Redo Records Processed - Count of redo records processed by SQL Apply<br>• Txns Delivered to Client - Count of SQL transactions processed by SQL Apply<br>• DML txns delivered - Count of DML transactions processed by SQL Apply<br>• DDL txns delivered - Count of DDL transactions processed by SQL Apply<br>• CTAS txns delivered - Count of CREATE TABLE AS SELECT (CTAS) transactions processed by SQL Apply<br>• Recursive txns delivered - Count of recursive transactions processed by SQL Apply<br>• Rolled back txns seen<br>• LCRs delivered to client - Number of logical change records (LCRs) processed by SQL Apply<br>• Bytes paged out - Cumulative count of bytes that have been paged out. LogMiner pages out memory from the LCR cache to accommodate certain ill-behaved workloads or under-configured systems. The ratio of bytes paged out to bytes of redo processed should be low. If this ratio is high (10% or higher), try increasing the MAX_SGA allocated to SQL Apply. |

| Column | Datatype | Description |
|---|---|---|
| | | • Microsecs spent in pageout - Time spent by LogMiner paging out memory from the LCR cache<br>• Bytes checkpointed - Keeps track of the amount of bytes checkpointed. The mining engine takes periodic checkpoints, whereby it writes out logical change records (LCRs) pertaining to long-running transactions. The ratio of `Bytes Checkpointed` to `Bytes of Redo Processed` should be low. A high ratio (10% or higher) indicates an ill-behaved workload.<br>• Microsecs spent in checkpoint - Time spent by the mining engine taking checkpoints, whereby it writes out logical change records (LCRs) pertaining to long-running transactions.<br>• Bytes rolled back - Cumulative value of the number of bytes rolled back by LogMiner. There are times that LogMiner needs to backtrack and reprocess a section of the redo stream. In this case, it will roll back work it has already done. The ratio of `Bytes Rolled Back` to `Bytes of Redo Processed` should be low. If this ratio is high (10% or higher), reduce the number of `PREPARER` processes allocated to SQL Apply.<br>• Microsecs spent in rollback - Time spent rolling back transactions already applied to the logical standby database |
| `VALUE` | `VARCHAR2(64)` | The corresponding metric value |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGSTDBY

`V$LOGSTDBY` is deprecated.

| Column | Datatype | Description |
|---|---|---|
| `SERIAL#` | `NUMBER` | SQL Session serial number. This data is used when joining this view with the `V$SESSION` and `V$PX_SESSION` views. |
| `LOGSTDBY_ID` | `NUMBER` | Parallel query slave ID |
| `PID` | `VARCHAR2(24)` | Process ID of the SQL apply process |
| `TYPE` | `VARCHAR2(30)` | Indicates the task being performed by the process (`COORDINATOR`, `APPLIER`, `ANALYZER`, `READER`, `PREPARER`, or `BUILDER`) |
| `STATUS_CODE` | `NUMBER` | Status number (or Oracle error code) belonging to the STATUS message |
| `STATUS` | `VARCHAR2(256)` | Description of the current activity of the process |

| Column | Datatype | Description |
|---|---|---|
| HIGH_SCN | NUMBER | Highest system change number (SCN) seen by the process. This column is used to confirm the progress of the individual process. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGSTDBY_PROCESS

V$LOGSTDBY_PROCESS displays dynamic information about what is happening to the Data Guard log apply services. This view is helpful when diagnosing performance problems during the logical application of archived redo logs to the standby database, and it can be helpful for other problems. This view is for logical standby databases only.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session id of the associated session. This matches the SID column of the corresponding row in the V$SESSION view. |
| SERIAL# | NUMBER | Serial number of the associated session. Together, (SID,SERIAL#) uniquely identify the session in the current database instance. |
| LOGSTDBY_ID | NUMBER | Parallel query slave ID |
| SPID | VARCHAR2(24) | This corresponds to the SPID value of the row corresponding to this process in the V$PROCESS view |
| TYPE | VARCHAR2(30) | Role that the process plays in the context of SQL Apply:<br>• COORDINATOR<br>• APPLIER<br>• ANALYZER<br>• READER<br>• PREPARER<br>• BUILDER |
| STATUS_CODE | NUMBER | Operation code identifying the current action of the process:<br>• 16111 - SQL Apply process is initializing<br>• 16112 - SQL Apply process is cleaning up as apply and mining processes are stopping based on a user command<br>• 16116 - SQL Apply process is idle<br>• 16117 - SQL Apply process is busy and is not waiting on any interesting event<br>• 16110 - APPLIER process has invoked a user-provided stored procedure in order to inspect a DDL statement prior to it being processed<br>• 16113 - APPLIER process is applying DML changes to some user object or to a sequence<br>• 16114 - APPLIER process is applying a DDL change<br>• 16115 - COORDINATOR process is loading the LogMiner dictionary from the redo stream |

| Column | Datatype | Description |
|---|---|---|
| | | • 16243 - BUILDER process is paging out memory to free up space in lcr cache<br>• 16240 - READER process idle waiting for additional logfile to be available<br>• 16241 - READER process is idle waiting for the logfile to fill the log sequence gap<br>• 16242 - READER process is processing a logfile |
| STATUS | VARCHAR2(256) | Description of the current action of the process |
| HIGH_SCN | NUMBER | Identifies the highest redo record/LCR processed by this process |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGSTDBY_PROGRESS

V$LOGSTDBY_PROGRESS displays the progress of log apply services on the logical standby database. This view is for logical standby databases only.

| Column | Datatype | Description |
|---|---|---|
| APPLIED_SCN | NUMBER | All the transactions with COMMIT SCN lower than or equal to this SCN have been applied |
| APPLIED_TIME | DATE | The time and date of APPLIED_SCN |
| RESTART_SCN | NUMBER | During an apply restart, LogMiner does not read any log file with a NEXT_CHANGE# lower than this SCN. |
| RESTART_TIME | DATE | The time and date of RESTART_SCN |
| LATEST_SCN | NUMBER | The highest SCN of all redo records that Logical Standby has encountered |
| LATEST_TIME | DATE | The time and date of LATEST_SCN |
| MINING_SCN | NUMBER | The SCN of the latest redo record processed by the builder process |
| MINING_TIME | DATE | The time and date of MINING_SCN |
| RESETLOGS_ID | NUMBER | A redo branch is identified by resetlogs SCN and resetlogs timestamp. The RESETLOGS_ID column contents are the same as resetlogs timestamp converted to a number. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$LOGSTDBY_STATE

V$LOGSTDBY_STATE provides consolidated information from V$LOGSTDBY and V$LOGSTDBY_STATS about the running state of Logical Standby.

| Column | Datatype | Description |
|---|---|---|
| PRIMARY_DBID | NUMBER | Database ID (DBID) of the primary database |
| PRIMARY_CON_DBID | NUMBER | This column indicates the DBID of the source database or source PDB corresponding to the database (or CDB) from which this column is queried.For a non-CDB and the root of a CDB, this column matches the PRIMARY_DBID column.For a maintained PDB, this column indicates the DBID for the corresponding PDB at the source.For a skipped PDB or a local PDB, this column would be NULL. |
| SESSION_ID | NUMBER | LogMiner session ID allocated to SQL Apply. |
| REALTIME_APPLY | VARCHAR2(64) | Y indicates that SQL Apply is running in real-time apply mode. If a standby redo log is configured, SQL Apply applies changes as they are written to the standby redo log files. N indicates that SQL Apply applies changes as each archived redo log file is received. |
| STATE | VARCHAR2(64) | • INITIALIZING: LogMiner session has been created and coordinator has attached to it<br>• LOADING DICTIONARY: SQL Apply is loading the LogMiner dictionary<br>• WAITING ON GAP: SQL Apply is waiting for a log file to be sent from the primary database<br>• APPLYING: SQL Apply is actively mining or applying transactions<br>• WAITING FOR DICTIONARY LOGS: SQL Apply is waiting for the archived logs containing the LogMiner dictionary to be shipped from the primary database<br>• IDLE: SQL Apply has applied all changes available at the logical standby, and is caught up with the primary database |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$LOGSTDBY"
- "V$LOGSTDBY_STATS"

# V$LOGSTDBY_STATS

V$LOGSTDBY_STATS displays statistics, current state, and status information related to SQL Apply. No rows are returned from this view when SQL Apply is not running. This view is only meaningful in the context of a logical standby database.

All statistics shown in this view are reinitialized at each SQL Apply start.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(64) | Name of the statistic, state, or status:<br>**Note:** Many of the following statistics are subject to change or deletion; programmers should write application code to tolerate missing or extra statistics.<br>• id of the logminer session used by SQL Apply to mine the redo logs.<br>• number of preparers<br>• number of appliers<br>• server processes in use for SQL Apply<br>• maximum SGA (in MBytes) for LCR cache<br>• whether SQL Apply is preserving the commit order seen at the primary database while applying changes<br>• maximum events recorded in the DBA_LOGSTDBY_EVENTS table<br>• whether SQL Apply is logging errors that are skipped in the DBA_LOGSTDBY_EVENTS table<br>• whether SQL Apply is logging DDLs that are skipped in the DBA_LOGSTDBY_EVENTS table<br>• whether SQL Apply is logging DDLs that are applied in the DBA_LOGSTDBY_EVENTS table<br>• whether SQL Apply is logging unsupported operations that are encountered in the DBA_LOGSTDBY_EVENTS table<br>• whether or not real time apply is on<br>• value of apply delay (in minutes)<br>• coordinator state<br>• coordinator uptime in seconds<br>• time of the most recent start of SQL Apply<br>• number of transactions mined and made available for apply<br>• number of transactions applied<br>• number of rolled back transactions mined<br>• number of DDL txns mined<br>• number of CTAS (Create Table as Select) txns mined<br>• number of thread enable events encountered in the redo stream<br>• number of thread disable events encountered in the redo stream<br>• bytes of redo records mined<br>• bytes paged out<br>• seconds spent in pageout activity |

| Column | Datatype | Description |
|---|---|---|
| | | • bytes checkpointed<br>• seconds spent in checkpointing activity<br>• seconds SQL Apply is idle<br>• number of times a complete standby redo logs are mined without having to mine the corresponding archived log<br>• number of times SQL Apply had to switch from a standby redo log to the corresponding archived log<br>• number of times SQL Apply mined redo from the archived logs<br>• number of archived logs that arrived at the standby via gap fetch mechanism (gap fetched logs mined)<br>• number of failed attempts to open a logfile<br>• amount of time spent in waiting for the current gap to resolve if SQL Apply is running in real time mode (current logfile wait)[1]<br>• time spent in waiting for gap to resolve if SQL Apply is running in real time mode (total logfile wait)[2] |
| VALUE | VARCHAR2(64) | Value of the statistic or state information |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

1 In case SQL Apply is not running in real time mode, this may not reflect time spent in gap resolution, but simply the time spent waiting for the most recent archived log to show up at the logical standby.
2 In case SQL Apply is not running in real time mode, this will include time that SQL Apply spent every time it finished processing all archived logs registered with it, and waited for the next log to be archived.

# V$LOGSTDBY_TRANSACTION

V$LOGSTDBY_TRANSACTION displays all transactions that are actively being processed by SQL Apply. The transaction identifiers shown in this view are those mined from the redo stream and correspond to transaction identifiers assigned at the primary database, and do not correspond to the transactions that are active at the logical standby database. For information regarding transactions active in the logical standby database, including those created as part of SQL Apply, query the V$TRANSACTION view at the logical standby database.

| Column | Datatype | Description |
|---|---|---|
| PRIMARY_XIDUSN | NUMBER | Undo segment number of the transaction |
| PRIMARY_XIDSLT | NUMBER | Slot number of the transaction |
| PRIMARY_XIDSQN | NUMBER | Sequence number of the transaction |
| PRIMARY_XID | RAW(8) | Transaction ID |
| PRIMARY_START_SCN | NUMBER | Start system change number (SCN) base |
| PRIMARY_START_TIME | DATE | Start time |

| Column | Datatype | Description |
|---|---|---|
| PRIMARY_PARENT_XID USN | NUMBER | Undo segment number of the parent transaction |
| PRIMARY_PARENT_XID SLT | NUMBER | Slot number of the parent transaction |
| PRIMARY_PARENT_XID SQN | NUMBER | Sequence number of the parent transaction |
| PRIMARY_PARENT_XID | RAW(8) | Transaction ID of the parent transaction (PDML) |
| TYPE | VARCHAR2(32) | Type:<br>• PL/SQL - Transaction was done as part of a supported PL/SQL procedure<br>• Direct Path Load - Transaction is a direct path load<br>• CTAS - Transaction contains at least one CREATE TABLE ... AS SELECT operation<br>• DDL - Transaction contains one or more DDL operations<br>• PDML Child - Transaction is a child transaction<br>• DML - Transaction contains only DML operations |
| MINING_STATUS | VARCHAR2(32) | Mining status:<br>• ACTIVE - Transaction is still being mined by LogMiner. At least part of this transaction is ready to be applied or has already been applied.<br>• COMPLETE - Transaction is complete and ready to be applied. LogMiner has finished mining. |
| SRC_CON_ID | NUMBER | Contains the PDB ID (the PDB_ID column from the DBA_PDBS view) of the source database that generated the change for this transaction. |
| APPLY_STATUS | VARCHAR2(6) | • ACTIVE - Transaction has been assigned to an apply server. It is in one of the following states:<br>- The transaction is being actively applied<br>- The transaction is being held by an apply server waiting for certain events to occur<br>- The transaction is being held by an apply server waiting for subsequent parts of this transaction<br>• NONE - Transaction has not yet been assigned to an apply server |
| SID | NUMBER | Session ID of the apply server's session; Null if APPLY_STATUS is NONE |
| SERIAL# | NUMBER | Serial number of the apply server's session; Null if APPLY_STATUS is NONE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$TRANSACTION"

# V$MANAGED_STANDBY

V$MANAGED_STANDBY displays current status information for some Oracle Database processes related to physical standby databases in the Data Guard environment. This view does not persist after an instance shutdown.

| Column | Datatype | Description |
|---|---|---|
| PROCESS | VARCHAR2(9) | Type of the process whose information is being reported:<br>• RFS - Remote file server<br>• MRP0 - Detached recovery server process<br>• MR(fg) - Foreground recovery session<br>• ARCH - Archiver process<br>• FGRD<br>• LGWR<br>• RFS(FAL)<br>• RFS(NEXP)<br>• LNS - ASYNC Redo Transport process |
| PID | VARCHAR2(24) | Operating system process identifier of the process |
| STATUS | VARCHAR2(12) | Current process status:<br>• UNUSED - No active process<br>• ALLOCATED - Process is active but not currently connected to a primary database<br>• CONNECTED - Network connection established to a primary database<br>• ATTACHED - Process is actively attached and communicating to a primary database<br>• IDLE - Process is not performing any activities<br>• ERROR - Process has failed<br>• OPENING - Process is opening the archived redo log<br>• CLOSING - Process has completed archival and is closing the archived redo log<br>• WRITING - Process is actively writing redo data to the archived redo log<br>• RECEIVING - Process is receiving network communication<br>• ANNOUNCING - Process is announcing the existence of a potential dependent archived redo log<br>• REGISTERING - Process is registering the existence of a completed dependent archived redo log<br>• WAIT_FOR_LOG - Process is waiting for the archived redo log to be completed<br>• WAIT_FOR_GAP - Process is waiting for the archive gap to be resolved<br>• APPLYING_LOG - Process is actively applying the archived redo log to the standby database |
| CLIENT_PROCESS | VARCHAR2(8) | Identifies the corresponding primary database process:<br>• Archival - Foreground (manual) archival process (SQL)<br>• ARCH - Background ARC*n* process |

| Column | Datatype | Description |
|---|---|---|
| | | • LGWR - Background LGWR process |
| CLIENT_PID | VARCHAR2(40) | Operating system process identifier of the client process |
| CLIENT_DBID | VARCHAR2(40) | Database identifier of the primary database |
| GROUP# | VARCHAR2(40) | Standby redo log group |
| RESETLOG_ID | NUMBER | Resetlogs identifier of the archived redo log |
| THREAD# | NUMBER | Archived redo log thread number |
| SEQUENCE# | NUMBER | Archived redo log sequence number |
| BLOCK# | NUMBER | Last processed archived redo log block number |
| BLOCKS | NUMBER | Count (in 512-byte blocks) of the last write to a redo log, or for a recovery process, the expected final read count |
| DELAY_MINS | NUMBER | Archived redo log delay interval in minutes |
| KNOWN_AGENTS | NUMBER | Total number of standby database agents processing an archived redo log |
| ACTIVE_AGENTS | NUMBER | Number of standby database agents actively processing an archived redo log |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_COMP_LIST

V$MAP_COMP_LIST displays supplementary information for all element mapping structures.

| Column | Datatype | Description |
|---|---|---|
| ELEM_IDX | NUMBER | Index corresponding to the element |
| NUM_COMP | NUMBER | Number of components (maximum is 5) |
| COMP1_NAME | VARCHAR2(256) | Name of the first component |
| COMP1_VAL | VARCHAR2(256) | Value of the first component |
| COMP2_NAME | VARCHAR2(256) | Name of the second component |
| COMP2_VAL | VARCHAR2(256) | Value of the second component |
| COMP3_NAME | VARCHAR2(256) | Name of the third component |
| COMP3_VAL | VARCHAR2(256) | Value of the third component |

| Column | Datatype | Description |
|---|---|---|
| COMP4_NAME | VARCHAR2(256) | Name of the fourth component |
| COMP4_VAL | VARCHAR2(256) | Value of the fourth component |
| COMP5_NAME | VARCHAR2(256) | Name of the fifth component |
| COMP5_VAL | VARCHAR2(256) | Value of the fifth component |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_ELEMENT

V$MAP_ELEMENT displays a list of all element mapping structures in the SGA of the instance.

| Column | Datatype | Description |
|---|---|---|
| ELEM_NAME | VARCHAR2(256) | Element name |
| ELEM_IDX | NUMBER | Index corresponding to the element |
| ELEM_CFGID | VARCHAR2(256) | Configuration ID (N/A if configuration ID is not supported) |
| ELEM_TYPE | VARCHAR2(12) | Element type:<br>• MIRROR<br>• STRIPE<br>• RAID5<br>• CONCATENATED<br>• PARTITION<br>• DISK<br>• NONE |
| ELEM_SIZE | NUMBER | Element Size in HKB |
| ELEM_NSUBELEM | NUMBER | Number of Subelements |
| ELEM_DESCR | VARCHAR2(256) | Element Description |
| STRIPE_SIZE | NUMBER | Stripe Size in HKB for RAID-5 and STRIPE elements, 0 for the remaining types |
| LIB_IDX | NUMBER | Index of the library which claims ownership of the element |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_EXT_ELEMENT

V$MAP_EXT_ELEMENT displays supplementary information for all element mapping structures.

| Column | Datatype | Description |
|---|---|---|
| ELEM_IDX | NUMBER | Index corresponding to the element |
| NUM_ATTRB | NUMBER | Number of Attributes (maximum is 5) |
| ATTRB1_NAME | VARCHAR2(256) | Name of the first Attribute |
| ATTRB1_VAL | VARCHAR2(256) | Value of the first attribute |
| ATTRB2_NAME | VARCHAR2(256) | Name of the second attribute |
| ATTRB2_VAL | VARCHAR2(256) | Value of the second attribute |
| ATTRB3_NAME | VARCHAR2(256) | Name of the third attribute |
| ATTRB3_VAL | VARCHAR2(256) | Value of the third attribute |
| ATTRB4_NAME | VARCHAR2(256) | Name of the fourth attribute |
| ATTRB4_VAL | VARCHAR2(256) | Value of the fourth attribute |
| ATTRB5_NAME | VARCHAR2(256) | Name of the fifth attribute |
| ATTRB5_VAL | VARCHAR2(256) | Value of the fifth attribute |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_FILE

V$MAP_FILE displays a list of all file mapping structures in the shared memory of the instance.

| Column | Datatype | Description |
|---|---|---|
| FILE_MAP_IDX | NUMBER | Index corresponding to the file |
| FILE_CFGID | VARCHAR2(256) | Configuration ID (N/A if configuration ID is not supported) |
| FILE_STATUS | VARCHAR2(7) | Status of the mapping information:<br>• VALID - File mapping information is latest<br>• INVALID - Mapping must be refreshed |

| Column | Datatype | Description |
|---|---|---|
| FILE_NAME | VARCHAR2(256) | Absolute file name |
| FILE_TYPE | VARCHAR2(11) | File type:<br>• DATAFILE<br>• SPFILE<br>• TEMPFILE<br>• CONTROLFILE<br>• LOGFILE<br>• ARCHIVEFILE |
| FILE_STRUCTURE | VARCHAR2(9) | File structure:<br>• FILE<br>• RAWVOLUME<br>• RAWDEVICE<br>• NONE |
| FILE_SIZE | NUMBER | File size in HKB (half KB) |
| FILE_NEXTS | NUMBER | Number of file extents in the file (not necessarily the same as the number of file extents mapped) |
| LIB_IDX | NUMBER | Index of mapping library claiming ownership of the file |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_FILE_EXTENT

V$MAP_FILE_EXTENT displays a list of all file extent mapping structures in the shared memory of the instance.

| Column | Datatype | Description |
|---|---|---|
| FILE_MAP_IDX | NUMBER | File index (corresponds to FILE_MAP_IDX in V$MAP_FILE) |
| EXT_NUM | NUMBER | File extent number |
| EXT_ELEM_OFF | NUMBER | Element offset in HKB |
| EXT_SIZE | NUMBER | File extent size in HKB |
| EXT_FILE_OFF | NUMBER | File Offset in HKB |
| EXT_TYPE | VARCHAR2(6) | File Extent Type:<br>• DATA<br>• PARITY<br>• NONE |
| ELEM_IDX | NUMBER | Index in V$MAP_ELEMENT corresponding to the element where the file extent resides |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_FILE_IO_STACK

V$MAP_FILE_IO_STACK displays the hierarchical arrangement of storage containers for files. Each row in the view represents a level in the hierarchy.

| Column | Datatype | Description |
|---|---|---|
| FILE_MAP_IDX | NUMBER | File index (corresponds to FILE_MAP_IDX in V$MAP_FILE) |
| DEPTH | NUMBER | Element depth within the I/O stack |
| ELEM_IDX | NUMBER | Index corresponding to the element |
| CU_SIZE | NUMBER | Contiguous set of logical blocks of the file (in HKB units) that is resident contiguously on the element |
| STRIDE | NUMBER | Number of HKB between contiguous units (CU) in the file that are contiguous on this element. Used in RAID5 and striped files. |
| NUM_CU | NUMBER | Number of contiguous units that are adjacent to each other on this element that are separated by STRIDE HKB in the file. In RAID5, the number of contiguous units also include the parity stripes. |
| ELEM_OFFSET | NUMBER | Element offset in HKB units |
| FILE_OFFSET | NUMBER | Offset in HKB units from the start of the file to the first byte of the contiguous units |
| DATA_TYPE | VARCHAR2(15) | Datatype:<br>• DATA<br>• PARITY<br>• DATA AND PARITY |
| PARITY_POS | NUMBER | Position of the parity. Only for RAID5. This column is needed to distinguish the parity from the data part. |
| PARITY_PERIOD | NUMBER | Parity period. Only for RAID5. |
| ID | NUMBER | Unique identifier |
| PARENT_ID | NUMBER | Parent identifier |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_LIBRARY

V$MAP_LIBRARY displays a list of all mapping libraries dynamically loaded by the external process.

| Column | Datatype | Description |
|---|---|---|
| LIB_IDX | NUMBER | Index corresponding to the library |
| LIB_NAME | VARCHAR2(256) | Absolute library name |
| VENDOR_NAME | VARCHAR2(64) | Name of the vendor implementing the library |
| PROTOCOL_NUM | NUMBER | Mapping protocol that the library supports |
| VERSION_NUM | VARCHAR2(32) | Version number |
| PATH_NAME | VARCHAR2(4000) | Path name |
| MAP_FILE | VARCHAR2(1) | Indicates whether the library supports mapping files (Y) or not (N) |
| FILE_CFGID | VARCHAR2(13) | Type of configuration ID supported for files:<br>• NONE - Not supported<br>• PERSISTENT<br>• NONPERSISTENT |
| MAP_ELEM | VARCHAR2(1) | Indicates whether the library supports mapping elements (Y) or not (N) |
| ELEM_CFGID | VARCHAR2(13) | Type of configuration id supported for elements:<br>• NONE - Not supported<br>• PERSISTENT<br>• NONPERSISTENT |
| MAP_SYNC | VARCHAR2(1) | Indicates whether the library needs to be explicitly synchronized so that future mappings reflect the most recent changes (Y) or not (N). Note that configuration IDs cannot be supported if the library needs to be explicitly synced. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAP_SUBELEMENT

V$MAP_SUBELEMENT displays a list of all subelement mapping structures in the shared memory of the instance.

| Column | Datatype | Description |
|---|---|---|
| CHILD_IDX | NUMBER | Index in V$MAP_ELEMENT corresponding to the child element |
| PARENT_IDX | NUMBER | Index in V$MAP_ELEMENT corresponding to the parent element |
| SUB_NUM | NUMBER | Subelement number |
| SUB_SIZE | NUMBER | Subelement size in HKB |
| ELEM_OFFSET | NUMBER | Offset in HKB on the child element |
| SUB_FLAGS | NUMBER | Subelement flags (currently unused) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MAPPED_SQL

V$MAPPED_SQL lists the SQL statements that are mapped to a different SQL statement for execution.

| Column | Datatype | Description |
|---|---|---|
| SQL_TEXT | VARCHAR2(1000) | First 1000 characters of the original SQL text |
| SQL_FULLTEXT | CLOB | Full text for the original SQL statement |
| SQL_ID | VARCHAR2(13) | SQL identifier of the original SQL statement |
| HASH_VALUE | NUMBER | Hash value of the original SQL statement |
| MAPPED_SQL_TEXT | VARCHAR2(1000) | First 1000 characters of the mapped SQL text |
| MAPPED_SQL_FULLTEXT | CLOB | Full text for the mapped SQL statement |
| MAPPED_SQL_ID | VARCHAR2(13) | SQL identifier of the mapped SQL statement |
| MAPPED_HASH_VALUE | NUMBER | Hash value of the mapped SQL statement |
| SQL_TRANSLATION_PROFILE_ID | NUMBER | A non-NULL value specifies the object ID of the SQL translation profile used to translate the SQL statement into the mapped SQL statement |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MEMORY_CURRENT_RESIZE_OPS

V$MEMORY_CURRENT_RESIZE_OPS displays information about memory resize operations (both automatic and manual) which are currently in progress. An operation can be a grow or a shrink of a dynamic memory component. All sizes are expressed in bytes.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT | VARCHAR2(64) | Component name |
| OPER_TYPE | VARCHAR2(13) | Operation type:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW<br>• SHRINK<br>• SHRINK_CANCEL |
| OPER_MODE | VARCHAR2(9) | Operation mode:<br>• MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| PARAMETER | VARCHAR2(80) | Name of the parameter for the resize operation |
| INITIAL_SIZE | NUMBER | Parameter value at the start of the operation |
| TARGET_SIZE | NUMBER | Desired value of the parameter after the resize |
| CURRENT_SIZE | NUMBER | Current value of the parameter |
| START_TIME | DATE | Start time of the operation |
| LAST_UPDATE_TIME | DATE | Last time progress was made for the operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MEMORY_DYNAMIC_COMPONENTS

V$MEMORY_DYNAMIC_COMPONENTS displays information about the dynamic SGA components. This view summarizes information based on all completed SGA resize operations since instance startup. All sizes are expressed in bytes.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT | VARCHAR2(64) | Component name |
| CURRENT_SIZE | NUMBER | Current size of the component |
| MIN_SIZE | NUMBER | Minimum size of the component since instance startup |
| MAX_SIZE | NUMBER | Maximum size of the component since instance startup |
| USER_SPECIFIED_SIZE | NUMBER | Value of the user parameter for the component |
| OPER_COUNT | NUMBER | Number of operations since instance startup |
| LAST_OPER_TYPE | VARCHAR2(13) | Last completed operation for the component:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW<br>• SHRINK<br>• SHRINK_CANCEL |
| LAST_OPER_MODE | VARCHAR2(9) | Mode of the last completed operation:<br>• MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| LAST_OPER_TIME | DATE | Start time of the last completed operation |
| GRANULE_SIZE | NUMBER | Granularity of the GROW or SHRINK operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MEMORY_RESIZE_OPS

V$MEMORY_RESIZE_OPS displays information about the last 800 completed memory resize operations (both automatic and manual). This does not include in-progress operations. All sizes are expressed in bytes.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT | VARCHAR2(64) | Component name |
| OPER_TYPE | VARCHAR2(13) | Operation type:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW<br>• SHRINK<br>• SHRINK_CANCEL |

| Column | Datatype | Description |
|---|---|---|
| OPER_MODE | VARCHAR2(9) | Operation mode:<br>• MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| PARAMETER | VARCHAR2(80) | Name of the parameter for the resize operation |
| INITIAL_SIZE | NUMBER | Parameter value at the start of the operation |
| TARGET_SIZE | NUMBER | Requested value of the parameter after the resize |
| FINAL_SIZE | NUMBER | Real value of the parameter after the resize |
| STATUS | VARCHAR2(9) | Completion status of the operation:<br>• INACTIVE<br>• PENDING<br>• COMPLETE<br>• CANCELLED<br>• ERROR |
| START_TIME | DATE | Start time of the operation |
| END_TIME | DATE | End time of the operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MEMORY_TARGET_ADVICE

V$MEMORY_TARGET_ADVICE provides information about how the MEMORY_TARGET parameter should be sized based on current sizing and satisfaction metrics.

| Column | Datatype | Description |
|---|---|---|
| MEMORY_SIZE | NUMBER | If the MEMORY_SIZE_FACTOR column has a value of 1, then this column shows the current size of memory, as set by the MEMORY_TARGET initialization parameter.<br>If the value of the MEMORY_SIZE_FACTOR column is less than or greater than 1, then this column shows a proposed memory size. |
| MEMORY_SIZE_FACTOR | NUMBER | A multiplier for the current memory size. Possible values are 0.25, 0.5, 0.75, 1, 1.5, 1.75, and 2. This multiplier times the current memory size equals the value of the MEMORY_SIZE column. |
| ESTD_DB_TIME | NUMBER | For current memory size (MEMORY_SIZE_FACTOR = 1), the amount of database time required to complete the current workload. For a proposed memory size, the estimated amount of database time that would be required if the MEMORY_TARGET parameter were changed to the proposed size. |

| Column | Datatype | Description |
|---|---|---|
| ESTD_DB_TIME_FACTOR | NUMBER | For a proposed memory size, ratio of estimated database time to current database time |
| VERSION | NUMBER | Version number of this recommendation (this snapshot of the V$MEMORY_TARGET_ADVICE view) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

Table 8-2 shows how the information provided in V$MEMORY_TARGET_ADVICE could be used to improve performance. The data indicates that if current memory size is 380M, and you were to increase it to 760M (2x), the current workload would take 80525 units of DBtime as opposed to 115475 units of DBtime, which is a significant improvement in performance.

***Table 8-2 Example of Using V$MEMORY_TARGET_ADVICE***

| MEMORY_SIZE | MEMORY_SIZE_FACTOR | ESTD_DB_TIME | ESTD_DB_TIME_FACTOR | VERSION |
|---|---|---|---|---|
| 380 | 1 | 115475 | 1 | 3 |
| 95 | .25 | 200500 | 1.7 | 3 |
| 190 | .5 | 125600 | 1.1 | 3 |
| 760 | 2 | 80525 | 0.7 | 3 |

**See Also:**

"MEMORY_TARGET"

# V$METRIC

V$METRIC displays the most recent statistic values for the complete set of metrics captured by the Automatic Workload Repository (AWR) infrastructure.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time for the metric value |
| END_TIME | DATE | End time for the metric value |
| INTSIZE_CSEC | NUMBER | Size of the time period |
| GROUP_ID | NUMBER | Metric Group ID. Refer to the V$METRICNAME view for the name of the group. |

| Column | Datatype | Description |
|---|---|---|
| ENTITY_ID | NUMBER | Entity ID for the metric in question. The value of the Entity ID depends upon the metric group. (See the following table for an explanation of possible values.) |
| ENTITY_SEQUENCE | NUMBER | Entity Sequence number for the metric in question. The value of the Entity Sequence depends upon the metric group. (See the following table for an explanation of possible values.) |
| METRIC_ID | NUMBER | Metric ID |
| METRIC_NAME | VARCHAR2(64) | Metric Name. This is the statistic that is captured for the entity. |
| VALUE | NUMBER | Value of the statistic between BEGIN_TIME and END_TIME |
| METRIC_UNIT | VARCHAR2(64) | Unit for the VALUE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

The following table describes what the ENTITY_ID and ENTITY_SEQUENCE are for each metric group:

| GID | Group NAME | Entity ID | Entity Sequence |
|---|---|---|---|
| 0 | Event Metrics | Event# | N/A |
| 1 | Event Class Metrics | Wait Class ID | N/A |
| 2 | System Metrics Long Duration | N/A | N/A |
| 3 | System Metrics Short Duration | N/A | N/A |
| 4 | Session Metrics Long Duration | Session ID | Serial# |
| 5 | Session Metrics Short Duration | Session ID | Serial# |
| 6 | Service Metrics | N/A | Service Hash |
| 7 | File Metrics Long Duration | File# | Creation Change# |
| 9 | Tablespace Metrics Long Duration | Tablespace# | N/A |
| 10 | Service Metrics (Short) | N/A | Service Hash |

# V$METRICGROUP

V$METRICGROUP displays information about the metric group for each of the four major Streams components: capture, propagation, apply, and queue.

| Column | Datatype | Description |
|---|---|---|
| GROUP_ID | NUMBER | Internal ID associated with each group |

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(64) | External name of the group |
| INTERVAL_SIZE | NUMBER | How often to collect statistics |
| MAX_INTERVAL | NUMBER | Total number of intervals over which statistics should be collected |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$METRIC_HISTORY

V$METRIC_HISTORY displays all the available statistic values for the complete set of metrics captured by the Automatic Workload Repository (AWR) infrastructure.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time for the metric value |
| END_TIME | DATE | End time for the metric value |
| INTSIZE_CSEC | NUMBER | Size of the time period |
| GROUP_ID | NUMBER | Metric Group ID. Refer to the V$METRICNAME view for the name of the group. |
| ENTITY_ID | NUMBER | Entity ID for the metric in question. The value of the Entity ID depends upon the metric group. (See the table in the description of V$METRIC for an explanation of possible values.) |
| ENTITY_SEQUENCE | NUMBER | Entity Sequence number for the metric in question. The value of the Entity Sequence depends upon the metric group. (See the table in the description of V$METRIC for an explanation of possible values.) |
| METRIC_ID | NUMBER | Metric ID |
| METRIC_NAME | VARCHAR2(64) | Metric Name. This is the statistic that is captured for the entity. |
| VALUE | NUMBER | Value of the statistic between BEGIN_TIME and END_TIME |
| METRIC_UNIT | VARCHAR2(64) | Unit for the VALUE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$METRICNAME

V$METRICNAME displays the mapping of the name of metrics to their metric ID.

| Column | Datatype | Description |
|---|---|---|
| GROUP_ID | NUMBER | Metric group ID |
| GROUP_NAME | VARCHAR2(64) | Metric group name |
| METRIC_ID | NUMBER | Metric ID |
| METRIC_NAME | VARCHAR2(64) | Metric name |
| METRIC_UNIT | VARCHAR2(64) | Unit of measurement |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MTTR_TARGET_ADVICE

V$MTTR_TARGET_ADVICE displays rows that predict the number of physical I/Os for the MTTR corresponding to each row. The rows also compute a physical I/O factor, which is the ratio of the number of estimated I/Os to the number of I/Os actually performed by the current MTTR setting during the measurement interval.

The content of the view is empty if MTTR advisory has not been turned on since database startup. Otherwise, it returns the advisory information collected. If advisory is currently off, then this information comes from the last time MTTR advisory was on. FAST_START_MTTR_TARGET must be set to a nonzero value if the STATISTICS_LEVEL parameter is dynamically modified to turn MTTR advisory on.

If the FAST_START_MTTR_TARGET parameter is changed while MTTR advisory is on, then MTTR advisory is temporarily turned off until the new FAST_START_MTTR_TARGET setting takes effect. During this transition period, the contents of V$MTTR_TARGET_ADVICE reflect the simulation result for the old MTTR setting.

| Column | Datatype | Description |
|---|---|---|
| MTTR_TARGET_FOR_ESTIMATE | NUMBER | MTTR setting being simulated. Equal to the current MTTR setting if this is the first row of the view. |
| ADVICE_STATUS | VARCHAR2(5) | Current status of MTTR simulation:<br>• ON<br>• READY<br>• OFF |
| DIRTY_LIMIT | NUMBER | Dirty buffer limit derived from the MTTR being simulated |
| ESTD_CACHE_WRITES | NUMBER | Estimated number of cache physical writes under this MTTR |

| Column | Datatype | Description |
|---|---|---|
| ESTD_CACHE_WRITE_FACTOR | NUMBER | Estimated cache physical write ratio under this MTTR. It is the ratio of the estimated number of cache writes to the number of cache writes under the current MTTR setting. |
| ESTD_TOTAL_WRITES | NUMBER | Estimated total number of physical writes under this MTTR |
| ESTD_TOTAL_WRITE_FACTOR | NUMBER | Estimated total physical write ratio under this MTTR. It is the ratio of the estimated total number of physical writes to the total number of physical writes under the current MTTR setting. |
| ESTD_TOTAL_IOS | NUMBER | Estimated total number of I/Os under this MTTR |
| ESTD_TOTAL_IO_FACTOR | NUMBER | Estimated total I/O ratio under this MTTR. It is the ratio of the estimated total number of I/Os to the total number of I/Os under the current MTTR setting. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "FAST_START_MTTR_TARGET"
- "STATISTICS_LEVEL"

# V$MUTEX_SLEEP

V$MUTEX_SLEEP shows the wait time, and the number of sleeps for each combination of mutex type and location.

| Column | Datatype | Description |
|---|---|---|
| MUTEX_TYPE | VARCHAR2(32) | Type of action/object the mutex protects |
| LOCATION | VARCHAR2(40) | The code location where the waiter slept for the mutex |
| SLEEPS | NUMBER | Number of sleeps for this MUTEX_TYPE and LOCATION |
| WAIT_TIME | NUMBER | Wait time in microseconds |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MUTEX_SLEEP_HISTORY

V$MUTEX_SLEEP_HISTORY displays time-series data. Each row in this view is for a specific time, mutex type, location, requesting session and blocking session combination. That is, it shows data related to a specific session (requesting session) that slept while requesting a specific mutex type and location, because it was being held by a specific blocking session. The data in this view is contained within a circular buffer, with the most recent sleeps shown.

| Column | Datatype | Description |
|---|---|---|
| MUTEX_IDENTIFIER | NUMBER | Mutex ID |
| SLEEP_TIMESTAMP | TIMESTAMP(6) | The last date/time this MUTEX_TYPE and LOCATION was slept for by the REQUESTING_SESSION, while being held by the BLOCKING_SESSION. |
| MUTEX_TYPE | VARCHAR2(32) | Type of action/object the mutex protects |
| GETS | NUMBER | The total number of gets since the mutex was created and up until the time of the wait (and from all sessions past and present) |
| SLEEPS | NUMBER | The number of times the requester had to sleep before obtaining the mutex |
| REQUESTING_SESSION | NUMBER | The SID of a session requesting the mutex |
| BLOCKING_SESSION | NUMBER | The SID of a session holding the mutex |
| LOCATION | VARCHAR2(40) | The code location where the waiter slept for the mutex |
| MUTEX_VALUE | RAW(4 \| 8) | If the mutex is held in exclusive (X) mode, this column shows the SID of the blocking session, else it shows the number of sessions referencing the mutex in S mode. |
| P1 | NUMBER | Reserved for internal use |
| P1RAW | RAW(4 \| 8) | Reserved for internal use |
| P2 | NUMBER | Reserved for internal use |
| P3 | NUMBER | Reserved for internal use |
| P4 | NUMBER | Reserved for internal use |
| P5 | VARCHAR2(64) | Reserved for internal use |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MVREFRESH

V$MVREFRESH displays information about the materialized views currently being refreshed.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |
| SERIAL# | NUMBER | Session serial number, which is used to uniquely identify a session's objects. Guarantees that session-level commands are applied to the correct session objects if the session ends with, and another session begins with, the same session ID. |
| CURRMVOWNER | VARCHAR2(31) | Owner of the materialized view currently being refreshed. The materialized view resides in this user's schema. |
| CURRMVNAME | VARCHAR2(31) | Name of the materialized view currently being refreshed |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$MYSTAT

V$MYSTAT contains statistics on the current session.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | ID of the current session |
| STATISTIC# | NUMBER | Number of the statistic |
| VALUE | NUMBER | Value of the statistic |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$NFS_CLIENTS

V$NFS_CLIENTS displays information about NFS clients currently connected to the XML DB NFS Server.

| Column | Datatype | Description |
|---|---|---|
| CLIENTID | NUMBER | A number identifying the client |

| Column | Datatype | Description |
|---|---|---|
| PRINCIPAL | VARCHAR2(2000) | User string denoting the principal that set the client ID (SetClientId) |
| CLIENTOPAQUEIDENTIFIER | VARCHAR2(1000) | Opaque string presented as identification by the client to the NFS server |
| VERIFIER | RAW(8) | Verifier presented by the client |
| LEASEEXPIRY | NUMBER | Number of seconds in which the lease expires for the client |
| CLIENTADDR | VARCHAR2(2000) | Address of the client |
| CONFIRMED | VARCHAR2(5) | TRUE if the client is confirmed; otherwise FALSE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$NFS_LOCKS

V$NFS_LOCKS displays information about byte range locks held on different files by NFS clients.

| Column | Datatype | Description |
|---|---|---|
| OPENSTATEID | RAW(16) | Open state ID of the open owner |
| OPENSEQUENCEID | NUMBER | Open Sequence ID of the open owner |
| LOCKSTATEID | RAW(16) | Lock state ID of the lock owner |
| LOCKSEQUENCEID | NUMBER | Lock sequence ID of the lock owner |
| LOCKOWNER | VARCHAR2(2000) | Opaque string presented as identification by the lock owner to the NFS server |
| OFFSET | NUMBER | Byte Offset from which lock starts |
| LENGTH | NUMBER | Length of the lock |
| LOCKTYPE | VARCHAR2(20) | Type of the lock |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$NFS_OPEN_FILES

V$NFS_OPEN_FILES displays information about all the files currently opened by clients at the NFS server.

| Column | Datatype | Description |
|---|---|---|
| CLIENTID | NUMBER | Number identifying the client |
| OPENOWNEROPAQUE | VARCHAR2(2000) | All the files currently opened by clients at the NFS server |
| OPENSTATEID | RAW(16) | Open state ID of the open owner |
| FILEHANDLE | RAW(32) | FileHandle of the file that has been opened |
| OPENSEQUENCEID | NUMBER | Open sequence ID of open owner |
| OPENREAD | VARCHAR2(5) | TRUE if the file is open for READ operations; otherwise FALSE |
| OPENWRITE | VARCHAR2(5) | TRUE if the file is open for WRITE operations; otherwise FALSE |
| SHAREACCESS | VARCHAR2(15) | Sharing mode of the file (SharedReadWrite, SharedRead, SharedWrite) |
| SHAREDENY | VARCHAR2(13) | Deny mode of the file (DenyReadWrite, DenyRead, DenyWrite) |
| CONFIRMED | VARCHAR2(5) | TRUE if open is confirmed; otherwise FALSE |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$NLS_PARAMETERS

V$NLS_PARAMETERS contains current values of NLS parameters.

| Column | Datatype | Description |
|---|---|---|
| PARAMETER | VARCHAR2(64) | Parameter names are as follows: NLS_CALENDAR, NLS_CHARACTERSET, NLS_COMP, NLS_CURRENCY, NLS_DATE_FORMAT, NLS_DATE_LANGUAGE, NLS_DUAL_CURRENCY, NLS_ISO_CURRENCY, NLS_LANGUAGE, NLS_LENGTH_SEMANTICS, NLS_NCHAR_CHARACTERSET, NLS_NCHAR_CONV_EXCP, NLS_NUMERIC_CHARACTERS, NLS_SORT, NLS_TERRITORY, NLS_TIMESTAMP_FORMAT, NLS_TIMESTAMP_TZ_FORMAT<br>Two additional parameters, NLS_TIME_FORMAT and NLS_TIME_TZ_FORMAT, are currently used for internal purposes only. |
| VALUE | VARCHAR2(64) | NLS parameter value |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$NLS_VALID_VALUES

V$NLS_VALID_VALUES lists all valid values for NLS parameters.

| Column | Datatype | Description |
|---|---|---|
| PARAMETER | VARCHAR2(64) | Parameter name (LANGUAGE \| SORT \| TERRITORY \| CHARACTERSET) |
| VALUE | VARCHAR2(64) | NLS parameter value |
| ISDEPRECATED | VARCHAR2(5) | Indicates whether the parameter has been deprecated (TRUE) or not (FALSE) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$NONLOGGED_BLOCK

V$NONLOGGED_BLOCK displays ranges of nonlogged datafile blocks recorded in the control file.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Absolute file number of the data file that contains the nonlogged blocks |
| BLOCK# | NUMBER | Block number of the first nonlogged block in the range of nologged blocks |
| BLOCKS | NUMBER | Number of nonlogged blocks found starting with BLOCK# |
| NONLOGGED_START_CHANGE# | NUMBER | The smallest SCN on which any block in this block range became nonlogged. NULL if unknown. |
| NONLOGGED_START_TIME | DATE | The time that corresponds to NONLOGGED_START_CHANGE#. NULL if unknown. |
| NONLOGGED_END_CHANGE# | NUMBER | The largest SCN on which any block in this block range became nonlogged. NULL if unknown. |

| Column | Datatype | Description |
|---|---|---|
| NONLOGGED_END_TIME | DATE | The time that corresponds to NONLOGGED_END_CHANGE#. NULL if unknown. |
| RESETLOGS_CHANGE# | NUMBER | The resetlogs SCN of the incarnation on which this block range was first marked as nonlogged. NULL if unknown. |
| RESETLOGS_TIME | DATE | The resetlogs time of the incarnation on which this block range was first marked as nologged. NULL if unknown. |
| OBJECT# | VARCHAR2(40) | The object ID this range belongs to. If this field is NULL, the object number is unknown. |
| REASON | CHAR(7) | The reason why this block range appears in this list. For Oracle Database 12*c*, it is always UNKNOWN. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$OBJECT_DEPENDENCY

V$OBJECT_DEPENDENCY displays the objects depended on by a package, procedure, or cursor that is currently loaded in the shared pool. For example, together with V$SESSION and V$SQL, this view can be used to determine which tables are used in the SQL statement that a user is currently executing.

**See Also:**

"V$SESSION" and "V$SQL"

| Column | Datatype | Description |
|---|---|---|
| FROM_ADDRESS | RAW(4 \| 8) | Address of a procedure, package, or cursor that is currently loaded in the shared pool |
| FROM_HASH | NUMBER | Hash value of a procedure, package, or cursor that is currently loaded in the shared pool |
| TO_OWNER | VARCHAR2(64) | Owner of the object that is depended on |
| TO_NAME | VARCHAR2(1000) | Name of the object that is depended on |
| TO_ADDRESS | RAW(4 \| 8) | Address of the object that is depended on. These can be used to look up more information on the object in V$DB_OBJECT_CACHE. |
| TO_HASH | NUMBER | Hash value of the object that is depended on. These can be used to look up more information on the object in V$DB_OBJECT_CACHE. |

| Column | Datatype | Description |
|---|---|---|
| TO_TYPE | NUMBER | Type of the object that is depended on |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$OBJECT_PRIVILEGE

V$OBJECT_PRIVILEGE displays information about privileges associated with an object.

| Column | Datatype | Description |
|---|---|---|
| OBJECT_TYPE_NAME | VARCHAR2(64) | Name of the object type |
| OBJECT_TYPE_ID | NUMBER | ID of the object type |
| PRIVILEGE_ID | NUMBER | ID of the privilege |
| PRIVILEGE_NAME | VARCHAR2(64) | Name of the privilege |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$OBJECT_USAGE

V$OBJECT_USAGE displays statistics about index usage gathered from the database for the indexes owned by the current user. You can use this view to monitor index usage. All indexes that have been used at least once can be monitored and displayed in this view.

**Note:**

The V$OBJECT_USAGE view is deprecated in Oracle Database 12*c* Release 1 (12.1) and maintained for backward compatibility. Support for this view may be removed in a future release. Oracle recommends that you use the USER_OBJECT_USAGE view instead of the V$OBJECT_USAGE view.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| INDEX_NAME | VARCHAR2 | NOT NULL | Index name in sys.obj$.name |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLE_NAME | VARCHAR2 | NOT NULL | Table name in sys.obj$.name |
| MONITORING | VARCHAR2 | | YES\| NO |
| USED | VARCHAR2 | | YES\| NO |
| START_MONITORING | VARCHAR2 | | Start monitoring time in sys.object_stats.start_monitoring |
| END_MONITORING | VARCHAR2 | | End monitoring time in sys.object_stats.end_monitoring |

**See Also:**

- "USER_OBJECT_USAGE"
- "DBA_OBJECT_USAGE"

# V$OBSOLETE_BACKUP_FILES

V$OBSOLETE_BACKUP_FILES displays all obsolete backups, copies, and archived logs according to the current retention policy. This view requires that the database is set using the DBMS_RCVMAN.SETDATABASE procedure.

| Column | Datatype | Description |
|---|---|---|
| PKEY | NUMBER | Primary key for the backup |
| BACKUP_TYPE | VARCHAR2(32) | Type of the backup:<br>• BACKUP SET<br>• COPY<br>• PROXY COPY |
| FILE_TYPE | VARCHAR2(32) | Type of the file:<br>• DATAFILE<br>• CONTROLFILE<br>• SPFILE<br>• REDO LOG<br>• PIECE |
| KEEP | VARCHAR2(3) | Indicates whether the backup has a retention policy different from the value for CONFIGURE RETENTION POLICY (YES) or not (NO) |
| KEEP_UNTIL | DATE | If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the date after which the backup becomes obsolete. If the column is null and KEEP_OPTIONS is not null, the backup never becomes obsolete. |
| KEEP_OPTIONS | VARCHAR2(13) | KEEP options for the backup:<br>• LOGS - RMAN keeps the logs needed to recover the backup<br>• NOLOGS - RMAN does not keep the logs needed to recover the backup |

| Column | Datatype | Description |
|---|---|---|
| | | If this column is null, then the backup has no KEEP options and will be made obsolete based on the retention policy. |
| STATUS | VARCHAR2(16) | Status of the backup:<br>• AVAILABLE<br>• UNAVAILABLE<br>• EXPIRED<br>• OTHER |
| FNAME | VARCHAR2(1024) | Name of the file |
| TAG | VARCHAR2(32) | Tag of the piece, copy, or proxy copy |
| MEDIA | VARCHAR2(80) | Media ID of the piece or proxy copy |
| RECID | NUMBER | Recid of the record in the controlfile |
| STAMP | NUMBER | Stamp of the record in the controlfile |
| DEVICE_TYPE | VARCHAR2(255) | Type of media device that stores the backup |
| BLOCK_SIZE | NUMBER | Block size for the backup (in bytes) |
| COMPLETION_TIME | DATE | Time when the backup completed |
| BS_KEY | NUMBER | Primary key of the backup set (valid only when BACKUP_TYPE is BACKUP SET) |
| BS_COUNT | NUMBER | Count of the backup set from the controlfile record (valid only when BACKUP_TYPE is BACKUP SET) |
| BS_STAMP | NUMBER | Stamp of the backup set from the controlfile record (valid only when BACKUP_TYPE is BACKUP SET) |
| BS_TYPE | VARCHAR2(32) | Type of the backup set (valid only when BACKUP_TYPE is BACKUP SET):<br>• DATAFILE<br>• ARCHIVED LOG |
| BS_INCR_TYPE | VARCHAR2(32) | Incremental level of the backup set (valid only when BACKUP_TYPE is BACKUP SET) |
| BS_PIECES | NUMBER | Number of backup pieces in the backup set (valid only when BACKUP_TYPE is BACKUP SET) |
| BS_COMPLETION_TIME | DATE | Completion time of the backup set (valid only when BACKUP_TYPE is BACKUP SET) |
| BP_PIECE# | NUMBER | Number of the backup piece (valid only when FILE_TYPE is PIECE and BACKUP_TYPE is BACKUP SET) |
| BP_COPY# | NUMBER | Copy number of the backup piece (valid only when FILE_TYPE is PIECE and BACKUP_TYPE is BACKUP SET) |
| DF_FILE# | NUMBER | Absolute file number of the datafile (valid only when FILE_TYPE is DATAFILE) |

| Column | Datatype | Description |
|---|---|---|
| DF_RESETLOGS_CHANGE# | NUMBER | System change number (SCN) of the most recent RESETLOGS when the control file or datafile was created (valid only when FILE_TYPE is DATAFILE) |
| DF_CREATION_CHANGE# | NUMBER | Creation SCN of the control file or datafile (valid only when FILE_TYPE is DATAFILE) |
| DF_CHECKPOINT_CHANGE# | NUMBER | System change number (SCN) of the most recent control file or datafile checkpoint (valid only when FILE_TYPE is DATAFILE) |
| DF_CKP_MOD_TIME | DATE | Modification time in case of SPFILE, otherwise time when the control file or datafile was checkpointed (valid only when FILE_TYPE is DATAFILE) |
| RL_THREAD# | NUMBER | Number of the redo thread (valid only when FILE_TYPE is REDO LOG) |
| RL_SEQUENCE# | NUMBER | Log sequence number (valid only when FILE_TYPE is REDO LOG) |
| RL_RESETLOGS_CHANGE# | NUMBER | System change number (SCN) of the most recent RESETLOGS when the record was created (valid only when FILE_TYPE is REDO LOG) |
| RL_FIRST_CHANGE# | NUMBER | First SCN of the redo log (valid only when FILE_TYPE is REDO LOG) |
| RL_FIRST_TIME | DATE | Time when Oracle switched into the redo log (valid only when FILE_TYPE is REDO LOG) |
| RL_NEXT_CHANGE# | NUMBER | First SCN of the next redo log in the thread (valid only when FILE_TYPE is REDO LOG) |
| RL_NEXT_TIME | DATE | First timestamp of the next redo log in the thread (valid only when FILE_TYPE is REDO LOG) |

**See Also:**

*Oracle Database Backup and Recovery User's Guide* for more information about the DBMS_RCVMAN.SETDATABASE procedure

# V$OBSOLETE_PARAMETER

V$OBSOLETE_PARAMETER displays information about obsolete initialization parameters. If any row of the view contains TRUE in the ISSPECIFIED column, then you should examine why.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(64) | Name of the parameter |
| ISSPECIFIED | VARCHAR2(5) | Indicates whether the parameter was specified in the parameter file (TRUE) or not (FALSE) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$OFFLINE_RANGE

V$OFFLINE_RANGE displays datafile offline information from the control file. Note that the last offline range of each datafile is kept in the DATAFILE record.

An offline range is created for a datafile when its tablespace is first altered to be OFFLINE NORMAL or READ ONLY, and then subsequently altered to be ONLINE or read/write. Note that no offline range is created if the datafile itself is altered to be OFFLINE or if the tablespace is altered to be OFFLINE IMMEDIATE.

**See Also:**

"V$DATAFILE"

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Record ID |
| STAMP | NUMBER | Record stamp |
| FILE# | NUMBER | Datafile number |
| OFFLINE_CHANGE# | NUMBER | SCN at which offlined |
| ONLINE_CHANGE# | NUMBER | SCN at which onlined |
| ONLINE_TIME | DATE | Time of offline SCN |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change number of the record |
| RESETLOGS_TIME | DATE | Resetlogs timestamp of the record |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$OPEN_CURSOR

V$OPEN_CURSOR lists cursors that each user session currently has opened and parsed, or cached.

| Column | Datatype | Description |
|---|---|---|
| SADDR | RAW(4 \| 8) | Session address |
| SID | NUMBER | Session identifier |
| USER_NAME | VARCHAR2(30) | User that is logged in to the session |
| ADDRESS | RAW(4 \| 8) | Used with HASH_VALUE to uniquely identify the SQL statement being executed in the session |
| HASH_VALUE | NUMBER | Used with ADDRESS to uniquely identify the SQL statement being executed in the session |
| SQL_ID | VARCHAR2(13) | SQL identifier of the SQL statement being executed in the session |
| SQL_TEXT | VARCHAR2(60) | First 60 characters of the SQL statement that is parsed into the open cursor |
| LAST_SQL_ACTIVE_TIME | DATE | Time when this cursor was last executed |
| SQL_EXEC_ID | NUMBER | If the open cursor is executing, then the SQL execution identifier for that execution (see V$SQL_MONITOR) |
| CURSOR_TYPE | VARCHAR2(64) | Type of cursor:<br>• OPEN PL/SQL - Open PL/SQL cursors<br>• OPEN - Other open cursors<br>• SESSION CURSOR CACHED - Cursors cached in the generic session cursor cache<br>• OPEN RECURSIVE - Open recursive cursors<br>• DICTIONARY LOOKUP CURSOR CACHED - Cursors cached in the dictionary lookup cursor cache<br>• BUNDLE DICTIONARY LOOKUP CACHED - Cursors cached in the bundled dictionary lookup cursor cache<br>• JAVA NAME TRANSLATION CURSOR CACHED - Cursors cached in the Java name translation cursor cache<br>• REPLICATION TRIGGER CURSOR CACHED - Cursors cached in the replication trigger cursor cache<br>• CONSTRAINTS CURSOR CACHED - Cursors cached in the constraints cursor cache<br>• PL/SQL CURSOR CACHED - Cursors cached in the PL/SQL cursor cache |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$OPTIMIZER_PROCESSING_RATE

V$OPTIMIZER_PROCESSING_RATE displays the processing rates used by the optimizer to compute degree of parallelism.

**Note:**

You can manipulate these rates using these procedures for the DBMS_STATS package:

- SET_PROCESSING_RATE
- DELETE_PROCESSING_RATE
- GATHER_PROCESSING_RATE

See *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4\| 8) | Address of the handle to the parent for this cursor |
| OPERATION_NAME | VARCHAR2(64) | Name of the operation. The possible values are AGGR, ALL, CPU, CPU_ACCESS, CPU_AGGR, CPU_BYTES_PER_SEC, CPU_FILTER, CPU_GBY, CPU_HASH_JOIN, CPU_JOIN, CPU_NL_JOIN, CPU_RANDOM_ACCESS, CPU_ROWS_PER_SEC, CPU_SEQUENTIAL_ACCESS, CPU_SM_JOIN, CPU_SORT, HASH, IO, IO_ACCESS, IO_BYTES_PER_SEC, IO_RANDOM_ACCESS, IO_ROWS_PER_SEC, IO_SEQUENTIAL_ACCESS, MEMCMP, MEMCPY |
| MANUAL_VALUE | VARCHAR2(10) | Value of the operation set manually by the user |
| CALIBRATION_VALUE | VARCHAR2(10) | Value of the operation obtained from calibration (by running the GATHER_PROCESSING_RATE procedure) |
| DEFAULT_VALUE | VARCHAR2(10) | Default value of the operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$OPTION

V$OPTION displays Oracle Database options and features. Typically, although not always, options must be separately licensed, whereas features come with the product and are enabled based on the product that is running (Standard Edition, Enterprise Edition, and so on).

| Column | Datatype | Description |
|---|---|---|
| PARAMETER | VARCHAR2(64) | Name of the option (or feature) |
| VALUE | VARCHAR2(64) | Indicates whether the option (or feature) is installed (TRUE) or not (FALSE) |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Licensing Information*

# V$OSSTAT

V$OSSTAT displays system utilization statistics from the operating system. One row is returned for each system statistic.

| Column | Datatype | Description |
|---|---|---|
| STAT_NAME | VARCHAR2(64) | Name of the statistic (see Table 8-3) |
| VALUE | NUMBER | Instantaneous statistic value |
| OSSTAT_ID | NUMBER | Statistic ID |
| COMMENTS | VARCHAR2(64) | Any additional operating system-specific clarifications for the statistic |
| CUMULATIVE | VARCHAR2(3) | Indicates whether the statistic is cumulative (that is, accumulates over time) (YES) or not (NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

***Table 8-3 V$OSSTAT Statistics***

| Statistic Name | Description |
|---|---|
| NUM_CPUS | Number of CPUs or processors available |
| IDLE_TIME | Number of hundredths of a second that a processor has been idle, totalled over all processors |
| BUSY_TIME | Number of hundredths of a second that a processor has been busy executing user or kernel code, totalled over all processors |
| USER_TIME | Number of hundredths of a second that a processor has been busy executing user code, totalled over all processors |

| Statistic Name | Description |
|---|---|
| SYS_TIME | Number of hundredths of a second that a processor has been busy executing kernel code, totalled over all processors |
| IOWAIT_TIME | Number of hundredths of a second that a processor has been waiting for I/O to complete, totalled over all processors |
| NICE_TIME | Number of hundredths of a second that a processor has been busy executing low-priority user code, totalled over all processors |
| AVG_IDLE_TIME | Number of hundredths of a second that a processor has been idle, averaged over all processors |
| AVG_BUSY_TIME | Number of hundredths of a second that a processor has been busy executing user or kernel code, averaged over all processors |
| AVG_USER_TIME | Number of hundredths of a second that a processor has been busy executing user code, averaged over all processors |
| AVG_SYS_TIME | Number of hundredths of a second that a processor has been busy executing kernel code, averaged over all processors |
| AVG_IOWAIT_TIME | Number of hundredths of a second that a processor has been waiting for I/O to complete, averaged over all processors |
| AVG_NICE_TIME | Number of hundredths of a second that a processor has been busy executing low-priority user code, averaged over all processors |
| OS_CPU_WAIT_TIME | Total number of hundredths of a second that processes have been in a ready state, waiting to be selected by the operating system scheduler to run |
| RSRC_MGR_CPU_WAIT_TIME | Total number of hundredths of a second that Oracle processes have been in a ready state, waiting for CPU to be available for their consumer group in the currently active resource plan |
| VM_IN_BYTES | Total number of bytes of data that have been paged in due to virtual memory paging |
| VM_OUT_BYTES | Total number of bytes of data that have been paged out due to virtual memory paging |
| PHYSICAL_MEMORY_BYTES | Total number of bytes of physical memory |
| LOAD | Current number of processes that are either running or in the ready state, waiting to be selected by the operating-system scheduler to run. On many platforms, this statistic reflects the average load over the past minute. |
| NUM_CPU_CORES | Number of CPU cores available (includes subcores of multicore CPUs as well as single-core CPUs) |
| NUM_CPU_SOCKETS | Number of CPU sockets available (represents an absolute count of CPU chips on the system, regardless of multithreading or multi-core architectures) |
| NUM_VCPUS | Number of virtual CPUs available |
| NUM_LCPUS | Number of logical CPUs available (includes hardware threads if hardware threading is turned on) |
| TCP_SEND_SIZE_MIN | Minimum size of the TCP send buffer |
| TCP_SEND_SIZE_DEFAULT | Default size of the TCP send buffer |

| Statistic Name | Description |
|---|---|
| TCP_SEND_SIZE_MAX | Maximum size of the TCP send buffer |
| TCP_RECEIVE_SIZE_MIN | Minimum size of the TCP receive buffer |
| TCP_RECEIVE_SIZE_DEFAULT | Default size of the TCP receive buffer |
| TCP_RECEIVE_SIZE_MAX | Maximum size of the TCP receive buffer |
| GLOBAL_SEND_SIZE_MAX | Maximum size of the global send buffer |
| GLOBAL_RECEIVE_SIZE_MAX | Maximum size of the global receive buffer |

**Note:**

The availability of all statistics except for NUM_CPUS and RSRC_MGR_CPU_WAIT_TIME is subject to the operating system platform on which the Oracle Database is running.

## V$PARALLEL_DEGREE_LIMIT_MTH

V$PARALLEL_DEGREE_LIMIT_MTH displays all available parallel degree limit resource allocation methods.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(40) | Name of the parallel degree limit resource allocation method |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$PARAMETER

V$PARAMETER displays information about the initialization parameters that are currently in effect for the session. A new session inherits parameter values from the instance-wide values displayed by the V$SYSTEM_PARAMETER view.

| Column | Datatype | Description |
|---|---|---|
| NUM | NUMBER | Parameter number |
| NAME | VARCHAR2(80) | Name of the parameter |
| TYPE | NUMBER | Parameter type:<br>• 1 - Boolean<br>• 2 - String<br>• 3 - Integer<br>• 4 - Parameter file<br>• 5 - Reserved |

| Column | Datatype | Description |
|---|---|---|
| | | • 6 - Big integer |
| VALUE | VARCHAR2(4000) | Parameter value for the session (if modified within the session); otherwise, the instance-wide parameter value |
| DISPLAY_VALUE | VARCHAR2(4000) | Parameter value in a user-friendly format. For example, if the VALUE column shows the value 262144 for a big integer parameter, then the DISPLAY_VALUE column will show the value 256K. |
| DEFAULT_VALUE | VARCHAR2(255) | The default value for this parameter |
| ISDEFAULT | VARCHAR2(9) | Indicates whether the parameter is set to the default value (TRUE) or the parameter value was specified in the parameter file (FALSE) |
| ISSES_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be changed with ALTER SESSION (TRUE) or not (FALSE) |
| ISSYS_MODIFIABLE | VARCHAR2(9) | Indicates whether the parameter can be changed with ALTER SYSTEM and when the change takes effect:<br>• IMMEDIATE - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect immediately.<br>• DEFERRED - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect in subsequent sessions.<br>• FALSE - Parameter cannot be changed with ALTER SYSTEM unless a server parameter file was used to start the instance. The change takes effect in subsequent instances. |
| ISPDB_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be modified inside a PDB (TRUE) or not (FALSE).<br>In a non-CDB, the value of this column is NULL. |
| ISINSTANCE_MODIFIABLE | VARCHAR2(5) | For parameters that can be changed with ALTER SYSTEM, indicates whether the value of the parameter can be different for every instance (TRUE) or whether the parameter must have the same value for all Real Application Clusters instances (FALSE). If the ISSYS_MODIFIABLE column is FALSE, then this column is always FALSE. |
| ISMODIFIED | VARCHAR2(10) | Indicates whether the parameter has been modified after instance startup:<br>• MODIFIED - Parameter has been modified with ALTER SESSION<br>• SYSTEM_MOD - Parameter has been modified with ALTER SYSTEM (which causes all the currently logged in sessions' values to be modified)<br>• FALSE - Parameter has not been modified after instance startup |
| ISADJUSTED | VARCHAR2(5) | Indicates whether Oracle adjusted the input value to a more suitable value (for example, the parameter value should be prime, but the user input a non-prime number, so Oracle adjusted the value to the next prime number) |

| Column | Datatype | Description |
|---|---|---|
| ISDEPRECATED | VARCHAR2(5) | Indicates whether the parameter has been deprecated (TRUE) or not (FALSE) |
| ISBASIC | VARCHAR2(5) | Indicates whether the parameter is a basic parameter (TRUE) or not (FALSE) |
| DESCRIPTION | VARCHAR2(255) | Description of the parameter |
| UPDATE_COMMENT | VARCHAR2(255) | Comments associated with the most recent update |
| HASH | NUMBER | Hash value for the parameter name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYSTEM_PARAMETER"

# V$PARAMETER_VALID_VALUES

V$PARAMETER_VALID_VALUES displays a list of valid values for list parameters.

| Column | Datatype | Description |
|---|---|---|
| NUM | NUMBER | Parameter number |
| NAME | VARCHAR2(64) | Parameter name |
| ORDINAL | NUMBER | Ordinal number in the list (1-based) |
| VALUE | VARCHAR2(255) | Parameter value at ordinal |
| ISDEFAULT | VARCHAR2(64) | Indicates whether the given ordinal value is the default value for the parameter |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PARAMETER2

V$PARAMETER2 displays information about the initialization parameters that are currently in effect for the session, with each list parameter value appearing as a row in

the view. A new session inherits parameter values from the instance-wide values displayed in the V$SYSTEM_PARAMETER2 view.

Presenting the list parameter values in this format enables you to quickly determine the values for a list parameter. For example, if a parameter value is a, b, then the V$PARAMETER view does not tell you if the parameter has two values (both a and b) or one value (a, b). V$PARAMETER2 makes the distinction between the list parameter values clear.

| Column | Datatype | Description |
|---|---|---|
| NUM | NUMBER | Parameter number |
| NAME | VARCHAR2(80) | Name of the parameter |
| TYPE | NUMBER | Parameter type:<br>• 1 - Boolean<br>• 2 - String<br>• 3 - Integer<br>• 4 - Parameter file<br>• 5 - Reserved<br>• 6 - Big integer |
| VALUE | VARCHAR2(4000) | Parameter value for the session (if modified within the session); otherwise, the instance-wide parameter value |
| DISPLAY_VALUE | VARCHAR2(4000) | Parameter value in a user-friendly format. For example, if the VALUE column shows the value 262144 for a big integer parameter, then the DISPLAY_VALUE column will show the value 256K. |
| ISDEFAULT | VARCHAR2(6) | Indicates whether the parameter is set to the default value (TRUE) or the parameter value was specified in the parameter file (FALSE) |
| ISSES_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be changed with ALTER SESSION (TRUE) or not (FALSE) |
| ISSYS_MODIFIABLE | VARCHAR2(9) | Indicates whether the parameter can be changed with ALTER SYSTEM and when the change takes effect:<br>• IMMEDIATE - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect immediately.<br>• DEFERRED - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect in subsequent sessions.<br>• FALSE - Parameter cannot be changed with ALTER SYSTEM unless a server parameter file was used to start the instance. The change takes effect in subsequent instances. |
| ISPDB_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be modified inside a PDB (TRUE) or not (FALSE).<br>In a non-CDB, the value of this column is NULL. |
| ISINSTANCE_MODIFIABLE | VARCHAR2(5) | For parameters that can be changed with ALTER SYSTEM, indicates whether the value of the parameter can be different for every instance (TRUE) or whether the parameter must have the same value for all Real Application Clusters instances (FALSE). If the ISSYS_MODIFIABLE column is FALSE, then this column is always FALSE. |

| Column | Datatype | Description |
|---|---|---|
| ISMODIFIED | VARCHAR2(10) | Indicates whether the parameter has been modified after instance startup:<br>• MODIFIED - Parameter has been modified with ALTER SESSION<br>• SYSTEM_MOD - Parameter has been modified with ALTER SYSTEM (which causes all the currently logged in sessions' values to be modified)<br>• FALSE - Parameter has not been modified after instance startup |
| ISADJUSTED | VARCHAR2(5) | Indicates whether Oracle adjusted the input value to a more suitable value (for example, the parameter value should be prime, but the user input a non-prime number, so Oracle adjusted the value to the next prime number) |
| ISDEPRECATED | VARCHAR2(5) | Indicates whether the parameter has been deprecated (TRUE) or not (FALSE) |
| ISBASIC | VARCHAR2(5) | Indicates whether the parameter is a basic parameter (TRUE) or not (FALSE) |
| DESCRIPTION | VARCHAR2(255) | Description of the parameter |
| ORDINAL | NUMBER | Position (ordinal number) of the parameter value. Useful only for parameters whose values are lists of strings. |
| UPDATE_COMMENT | VARCHAR2(255) | Comments associated with the most recent update |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SYSTEM_PARAMETER"
- "V$PARAMETER"

# V$PATCHES

V$PATCHES shows the patches applied on an Oracle ASM instance and the list of patches applied to an Oracle Grid infrastructure home directory.

**Note:**

If this view is queried in an RDBMS instance, no patch information is returned.

| Column | Datatype | Description |
|---|---|---|
| PATCH_ID | NUMBER | Patch identifier |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Grid Infrastructure Installation Guide for Linux* for information about applying Oracle ASM one-off patches.

# V$PDB_INCARNATION

V$PDB_INCARNATION displays information about all PDB incarnations. Oracle creates a new PDB incarnation whenever a PDB is opened with the RESETLOGS option.

| Column | Datatype | Description |
|---|---|---|
| DB_INCARNATION# | NUMBER | Database incarnation number |
| PDB_INCARNATION# | NUMBER | PDB incarnation number |
| STATUS | VARCHAR2(7) | Incarnation status:<br>• ORPHAN: Orphan incarnation<br>• CURRENT: Current incarnation of the PDB<br>• PARENT: Parent of the current incarnation |
| INCARNATION_SCN | NUMBER | The SCN to flashback or recover to for this PDB incarnation |
| INCARNATION_TIME | DATE | The point in time recovered to for this PDB incarnation |
| BEGIN_RESETLOGS_SCN | NUMBER | The SCN at the beginning of PDB resetlogs |
| BEGIN_RESETLOGS_TIME | DATE | The time at the beginning of PDB resetlogs |
| END_RESETLOGS_SCN | NUMBER | The SCN at the end of PDB resetlogs |
| END_RESETLOGS_TIME | DATE | The time at the end of PDB resetlogs |
| PRIOR_DB_INCARNATION# | NUMBER | Parent database incarnation number |
| PRIOR_PDB_INCARNATION# | VARCHAR2(40) | Parent PDB incarnation number |
| FLASHBACK_DATABASE_ALLOWED | VARCHAR2(3) | Indicates whether the PDB can be flashbacked to this incarnation as part of a flashback database operation for the CDB |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PDBS

V$PDBS displays information about PDBs associated with the current instance.

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| DBID | NUMBER | PDB identifier calculated when the PDB is created and stored in all file headers associated with the PDB |
| CON_UID | NUMBER | Unique identifier associated with the PDB |
| GUID | RAW(16) | Globally unique identifier (GUID) of this PDB |
| NAME | VARCHAR2(30) | Name of the PDB |
| OPEN_MODE | VARCHAR2(10) | Open mode information. Possible values:<br>• MOUNTED<br>• READ WRITE<br>• READ ONLY<br>• MIGRATE |
| RESTRICTED | VARCHAR2(3) | Indicates whether only users possessing RESTRICTED SESSION privilege can connect to the PDB. Possible values:<br>• YES<br>• NO<br>• NULL |
| OPEN_TIME | TIMESTAMP(3) | Date and time when the database was last opened |
| CREATE_SCN | NUMBER | System change number (SCN) for the creation of this PDB |
| TOTAL_SIZE | NUMBER | Disk space (in bytes) used by the container, including both data and temp files. |
| BLOCK_SIZE | NUMBER | The current block size for the PDB |
| RECOVERY_STATUS[1] | VARCHAR2(8) | Shows whether recovery is enabled or disabled for the PDB. Possible values:<br>• ENABLED |

| Column | Datatype | Description |
|---|---|---|
| | | • DISABLED |
| SNAPSHOT_PARENT_CON_ID[1] | NUMBER | This column shows the container ID of the master PDB that this PDB is a snapshot clone of. This column shows a non-zero value only if the PDB is a snapshot clone. For all other cases, it shows a value of 0. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

# V$PERSISTENT_PUBLISHERS

V$PERSISTENT_PUBLISHERS displays information about all active publishers of the persistent queues in the database. There is one row per instance per queue per publisher. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Identifier for the queue |
| QUEUE_SCHEMA | VARCHAR2(30) | Owner of the queue |
| QUEUE_NAME | VARCHAR2(30) | Name of the queue |
| PUBLISHER_NAME | VARCHAR2(30) | Name of the agent enqueuing the message |
| PUBLISHER_ADDRESS | VARCHAR2(1024) | Address of the publisher agent |
| PROTOCOL | NUMBER | Protocol used by the publisher's address |
| ENQUEUED_MSGS | NUMBER | Number of messages that have been enqueued |
| ELAPSED_ENQUEUE_TIME | NUMBER | Total time spent doing enqueue (in hundredths of a second) |
| ENQUEUE_CPU_TIME | NUMBER | Total CPU time for enqueue (in hundredths of a second) |
| LAST_ENQUEUE_TIME | TIMESTAMP(6) | Last enqueue message timestamp |
| ENQUEUE_TRANSACTIONS | NUMBER | Number of enqueue transactions |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PERSISTENT_QMN_CACHE

V$PERSISTENT_QMN_CACHE displays detailed information and statistics about the background activities for all queue tables in the system. There is one row per queue

table. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_TABLE_ID | NUMBER | Queue table object ID |
| TYPE | VARCHAR2(32) | Type of the queue table's queue monitor cache |
| STATUS | NUMBER | Status of the queue table's queue monitor cache |
| NEXT_SERVICE_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the queue table should be serviced by QMON servers |
| WINDOW_END_TIME | TIMESTAMP(3) WITH TIME ZONE | Time manager activity period for non-owner queue table operations |
| TOTAL_RUNS | NUMBER | Total number of times this queue table is served |
| TOTAL_LATENCY | NUMBER | Cumulative latency in serving the queue table (in hundredths of a second) |
| TOTAL_ELAPSED_TIME | NUMBER | Total time spent in processing this queue table (in seconds) |
| TOTAL_CPU_TIME | NUMBER | Cumulative CPU time for serving the queue table (in hundredths of a second) |
| TMGR_ROWS_PROCESSED | NUMBER | Number of time manager entries processed |
| TMGR_ELAPSED_TIME | NUMBER | Cumulative time for time management activities (in hundredths of a second) |
| TMGR_CPU_TIME | NUMBER | Cumulative CPU time for time management activities (in hundredths of a second) |
| LAST_TMGR_PROCESSING_TIME | TIMESTAMP(3) WITH TIME ZONE | Last timer manager processing time |
| DEQLOG_ROWS_PROCESSED | NUMBER | Number of dequeue log entries processed |
| DEQLOG_PROCESSING_ELAPSED_TIME | NUMBER | Total time for processing dequeue log entries (in hundredths of a second) |
| DEQLOG_PROCESSING_CPU_TIME | NUMBER | Total CPU time for processing dequeue log entries (in hundredths of a second) |
| LAST_DEQLOG_PROCESSING_TIME | TIMESTAMP(3) WITH TIME ZONE | Last dequeue log processing time |
| DEQUEUE_INDEX_BLOCKS_FREED | NUMBER | Number of dequeue index blocks freed |
| HISTORY_INDEX_BLOCKS_FREED | NUMBER | Number of history index blocks freed |

| Column | Datatype | Description |
|---|---|---|
| TIME_INDEX_BLOCKS_FREED | NUMBER | Number of time manager index blocks freed |
| INDEX_CLEANUP_COUNT | NUMBER | Number of times index block cleanup was attempted |
| INDEX_CLEANUP_ELAPSED_TIME | NUMBER | Total time for index block cleanup (in hundredths of a second) |
| INDEX_CLEANUP_CPU_TIME | NUMBER | Total CPU time for index block cleanup (in hundredths of a second) |
| LAST_INDEX_CLEANUP_TIME | TIMESTAMP(3) WITH TIME ZONE | Last index block cleanup time |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PERSISTENT_QUEUES

V$PERSISTENT_QUEUES displays information about all active persistent queues in the database since the queues' first activity time. There is one row per queue. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Identifier for the queue |
| QUEUE_TABLE_ID | NUMBER | Queue table identifier |
| QUEUE_SCHEMA | VARCHAR2(30) | Owner of the queue |
| QUEUE_NAME | VARCHAR2(30) | Name of the queue |
| FIRST_ACTIVITY_TIME | TIMESTAMP(6) | First queue activity time since database startup |
| ENQUEUED_MSGS | NUMBER | Number of messages enqueued |
| DEQUEUED_MSGS | NUMBER | Number of messages dequeued<br>**Note:** This column will not be incremented until all the subscribers of the message have dequeued the message and its retention time has elapsed. |
| BROWSED_MSGS | NUMBER | Number of messages that have been browsed |
| ELAPSED_ENQUEUE_TIME | NUMBER | Total time (in hundredths of a second) spent doing enqueue |
| ELAPSED_DEQUEUE_TIME | NUMBER | Total time (in hundredths of a second) spent doing dequeue |

| Column | Datatype | Description |
|---|---|---|
| ENQUEUE_CPU_TIME | NUMBER | Total CPU time for enqueue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | Total CPU time for dequeue (in hundredths of a second) |
| AVG_MSG_AGE | NUMBER | Average age of messages in the queue |
| DEQUEUED_MSG_LATENCY | NUMBER | Last dequeued message latency (in seconds) |
| ELAPSED_TRANSFORMATION_TIME | NUMBER | Total time (in hundredths of a second) spent doing transformation |
| ELAPSED_RULE_EVALUATION_TIME | NUMBER | Total time (in hundredths of a second) spent doing rule evaluation |
| ENQUEUED_EXPIRY_MSGS | NUMBER | Number of messages enqueued with expiry |
| ENQUEUED_DELAY_MSGS | NUMBER | Number of messages enqueued with delay |
| MSGS_MADE_EXPIRED | NUMBER | Number of messages expired by time manager |
| MSGS_MADE_READY | NUMBER | Number of messages made ready by time manager |
| LAST_ENQUEUE_TIME | TIMESTAMP(6) | Last message enqueue time |
| LAST_DEQUEUE_TIME | TIMESTAMP(6) | Last message dequeue time |
| LAST_TM_EXPIRY_TIME | TIMESTAMP(6) | Last time message was expired by time manager |
| LAST_TM_READY_TIME | TIMESTAMP(6) | Last time message was made ready by time manager |
| ENQUEUE_TRANSACTIONS | NUMBER | Number of enqueue transactions |
| DEQUEUE_TRANSACTIONS | NUMBER | Number of dequeue transactions |
| EXECUTION_COUNT | NUMBER | Number of executions of the dequeue cursor |
| OLDEST_MSGID | RAW(16) | Message ID of the oldest message in the queue |
| OLDEST_MSG_ENQTM | TIMESTAMP(6) | Enqueue time of the oldest message in the queue |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PERSISTENT_SUBSCRIBERS

V$PERSISTENT_SUBSCRIBERS displays information about all active subscribers of the persistent queues in the database. There is one row per instance per queue per

subscriber. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Identifier for the queue |
| QUEUE_SCHEMA | VARCHAR2(30) | Owner of the queue |
| QUEUE_NAME | VARCHAR2(30) | Name of the queue |
| SUBSCRIBER_ID | NUMBER | Internal subscriber number |
| SUBSCRIBER_NAME | VARCHAR2(30) | Name of the subscriber |
| SUBSCRIBER_ADDRESS | VARCHAR2(1024) | Address of the subscribing agent |
| PROTOCOL | NUMBER | Protocol of the subscribing agent |
| SUBSCRIBER_TYPE | VARCHAR2(30) | Type of the subscriber:<br>• PROXY - Propagation subscriber<br>• SUBSCRIBER - Normal subscriber<br>• RECIPIENT - Recipient |
| FIRST_ACTIVITY_TIME | TIMESTAMP(6) | First subscriber activity time since database startup |
| ENQUEUED_MSGS | NUMBER | Number of messages enqueued since FIRST_ACTIVITY_TIME |
| DEQUEUED_MSGS | NUMBER | Number of messages dequeued since FIRST_ACTIVITY_TIME |
| AVG_MSG_AGE | NUMBER | Average age of messages in the queue |
| BROWSED_MSGS | NUMBER | Number of messages that have been browsed |
| EXPIRED_MSGS | NUMBER | Number of messages expired since FIRST_ACTIVITY_TIME |
| DEQUEUED_MSG_LATENCY | NUMBER | Last dequeued message latency (in seconds) |
| LAST_ENQUEUE_TIME | TIMESTAMP(6) | Timestamp of the last enqueued message |
| LAST_DEQUEUE_TIME | TIMESTAMP(6) | Timestamp of the last dequeued message |
| ELAPSED_DEQUEUE_TIME | NUMBER | Total time spent in dequeue (in hundredths of a second) |
| DEQUEUE_CPU_TIME | NUMBER | Total CPU time for dequeue (in hundredths of a second) |
| DEQUEUE_TRANSACTIONS | NUMBER | Number of dequeue transactions |
| EXECUTION_COUNT | NUMBER | Number of executions of the dequeue index cursor |
| DEQUEUE_MEMORY_LOCKS | NUMBER | Number of dequeue transactions that obtained memory locks |
| DEQUEUE_DISK_LOCKS | NUMBER | Number of dequeue transactions that obtained disk locks |
| DEQUEUE_DISK_DELETES | NUMBER | Number of dequeue transactions that deleted index-organized table entries |

| Column | Datatype | Description |
|---|---|---|
| OLDEST_MSGID | RAW(16) | Message ID of the oldest message |
| OLDEST_MSG_ENQTM | TIMESTAMP(6) | Enqueue time of the oldest message |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PGA_TARGET_ADVICE

V$PGA_TARGET_ADVICE predicts how the cache hit percentage and over allocation count statistics displayed by the V$PGASTAT performance view would be impacted if the value of the PGA_AGGREGATE_TARGET parameter is changed. The prediction is performed for various values of the PGA_AGGREGATE_TARGET parameter, selected around its current value. The advice statistic is generated by simulating the past workload run by the instance.

The content of the view is empty if PGA_AGGREGATE_TARGET is not set. In addition, the content of this view is not updated if the STATISTICS_LEVEL parameter is set to BASIC. Base statistics for this view are reset at instance startup and when the value of the PGA_AGGREGATE_TARGET initialization parameter is dynamically modified.

| Column | Datatype | Description |
|---|---|---|
| PGA_TARGET_FOR_ESTIMATE | NUMBER | Value of PGA_AGGREGATE_TARGET for this prediction (in bytes) |
| PGA_TARGET_FACTOR | NUMBER | PGA_TARGET_FOR_ESTIMATE / the current value of the PGA_AGGREGATE_TARGET parameter |
| ADVICE_STATUS | VARCHAR2(3) | Indicates whether the advice is enabled (ON) or disabled (OFF) depending on the value of the STATISTICS_LEVEL parameter |
| BYTES_PROCESSED | NUMBER | Total bytes processed by all the work areas considered by this advice (in bytes) |
| ESTD_TIME | NUMBER | Time (in seconds) required to process the bytes |
| ESTD_EXTRA_BYTES_RW | NUMBER | Estimated number of extra bytes which would be read or written if PGA_AGGREGATE_TARGET was set to the value of the PGA_TARGET_FOR_ESTIMATE column. This number is derived from the estimated number and size of work areas which would run in one-pass (or multi-pass) for that value of PGA_AGGREGATE_TARGET. |
| ESTD_PGA_CACHE_HIT_PERCENTAGE | NUMBER | Estimated value of the cache hit percentage statistic when PGA_AGGREGATE_TARGET equals PGA_TARGET_FOR_ESTIMATE. This column is derived from the above two columns and is equal to BYTES_PROCESSED / (BYTES_PROCESSED + ESTD_EXTRA_BYTES_RW) |

| Column | Datatype | Description |
|---|---|---|
| ESTD_OVERALLOC_COUNT | NUMBER | Estimated number of PGA memory over-allocations if the value of PGA_AGGREGATE_TARGET is set to PGA_TARGET_FOR_ESTIMATE. A nonzero value means that PGA_TARGET_FOR_ESTIMATE is not large enough to run the work area workload. Hence, the DBA should not set PGA_AGGREGATE_TARGET to PGA_TARGET_FOR_ESTIMATE since Oracle will not be able to honor that target. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$PGASTAT"
- "PGA_AGGREGATE_TARGET"
- "STATISTICS_LEVEL"
- *Oracle Database Performance Tuning Guide* for information on tuning the PGA_AGGREGATE_TARGET initialization parameter using the PGA advice views

# V$PGA_TARGET_ADVICE_HISTOGRAM

V$PGA_TARGET_ADVICE_HISTOGRAM predicts how statistics displayed by the V$SQL_WORKAREA_HISTOGRAM dynamic view would be impacted if the value of the PGA_AGGREGATE_TARGET parameter is changed. This prediction is performed for various values of the PGA_AGGREGATE_TARGET parameter, selected around its current value. The advice statistic is generated by simulating the past workload run by the instance.

The content of the view is empty if PGA_AGGREGATE_TARGET is not set. In addition, the content of this view is not updated when the STATISTICS_LEVEL initialization parameter is set to BASIC. Base statistics for this view are reset at instance startup or when the value of the PGA_AGGREGATE_TARGET initialization parameter is dynamically modified.

| Column | Datatype | Description |
|---|---|---|
| PGA_TARGET_FOR_ESTIMATE | NUMBER | Value of PGA_AGGREGATE_TARGET for this prediction (in bytes) |
| PGA_TARGET_FACTOR | NUMBER | PGA_TARGET_FOR_ESTIMATE / the current value of the PGA_AGGREGATE_TARGET parameter |
| ADVICE_STATUS | VARCHAR2(3) | Indicates whether the advice is enabled (ON) or disabled (OFF) depending on the value of the STATISTICS_LEVEL parameter |

| Column | Datatype | Description |
|---|---|---|
| LOW_OPTIMAL_SIZE | NUMBER | Lower bound for the optimal memory requirement of work areas included in this row (in bytes) |
| HIGH_OPTIMAL_SIZE | NUMBER | Upper bound for the optimal memory requirement of work areas included in this row (in bytes) |
| ESTD_OPTIMAL_EXECUTIONS | NUMBER | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which are predicted to run optimal given a value of PGA_AGGREGATE_TARGET equal to PGA_TARGET_FOR_ESTIMATE |
| ESTD_ONEPASS_EXECUTIONS | NUMBER | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which are predicted to run one-pass given a value of PGA_AGGREGATE_TARGET equal to PGA_TARGET_FOR_ESTIMATE |
| ESTD_MULTIPASSES_EXECUTIONS | NUMBER | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which are predicted to run multi-pass given a value of PGA_AGGREGATE_TARGET equal to PGA_TARGET_FOR_ESTIMATE |
| ESTD_TOTAL_EXECUTIONS | NUMBER | Sum of ESTD_OPTIMAL_EXECUTIONS, ESTD_ONEPASS_EXECUTIONS, and ESTD_MULTIPASSES_EXECUTIONS |
| IGNORED_WORKAREAS_COUNT | NUMBER | Number of work areas with optimal memory requirement between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE ignored in the advice generation due to memory and CPU constraints |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SQL_WORKAREA_HISTOGRAM"
- "PGA_AGGREGATE_TARGET"
- "STATISTICS_LEVEL"
- *Oracle Database Performance Tuning Guide* for information on tuning the PGA_AGGREGATE_TARGET initialization parameter using the PGA advice views

# V$PGASTAT

V$PGASTAT displays PGA memory usage statistics as well as statistics about the automatic PGA memory manager when it is enabled (that is, when PGA_AGGREGATE_TARGET is set). Cumulative values in V$PGASTAT are accumulated since instance startup.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(64) | Name of the statistic (see Table 8-4) |
| VALUE | NUMBER | Statistic value |
| UNIT | VARCHAR2(12) | Unit for the value:<br>• bytes<br>• microseconds<br>• percent |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

***Table 8-4 V$PGASTAT Statistics***

| Statistic Name | Description |
|---|---|
| aggregate PGA auto target | Amount of PGA memory the Oracle Database can use for work areas running in automatic mode. This amount is dynamically derived from the value of the PGA_AGGREGATE_TARGET initialization parameter and the current work area workload, and continuously adjusted by the Oracle Database.<br>If this value is small compared to the value of PGA_AGGREGATE_TARGET, then a large amount of PGA memory is used by other components of the system (for example, PL/SQL or Java memory) and little is left for work areas. The DBA must ensure that enough PGA memory is left for work areas running in automatic mode. |
| aggregate PGA target parameter | Current value of the PGA_AGGREGATE_TARGET initialization parameter. If this parameter is not set, then its value is 0 and automatic management of PGA memory is disabled. |
| bytes processed | Number of bytes processed by memory intensive SQL operators, cumulated since instance startup. |
| cache hit percentage | A metric computed by the Oracle Database to reflect the performance of the PGA memory component, cumulative since instance startup. A value of 100% means that all work areas executed by the system since instance startup have used an optimal amount of PGA memory.<br>When a work area cannot run optimal, one or more extra passes is performed over the input data. This will reduce the cache hit percentage in proportion to the size of the input data and the number of extra passes performed. |

| Statistic Name | Description |
|---|---|
| extra bytes read/written | Number of bytes processed during extra passes of the input data, cumulated since instance startup. When a work area cannot run optimal, one or more of these extra passes is performed. |
| global memory bound | Maximum size of a work area executed in automatic mode. This value is continuously adjusted by the Oracle Database to reflect the current state of the work area workload. The global memory bound generally decreases when the number of active work areas is increasing in the system.<br>If the value of the global bound decreases below 1 MB, then the value of PGA_AGGREGATE_TARGET should be increased. |
| max processes count | Maximum number of processes active at any one time since instance startup. |
| maximum PGA allocated | Maximum number of bytes of PGA memory allocated at one time since instance startup. |
| maximum PGA used for auto workareas | Maximum amount of PGA memory consumed at one time by work areas running under the automatic memory management mode since instance startup. |
| maximum PGA used for manual workareas | Maximum amount of PGA memory consumed at one time by work areas running under the manual memory management mode since instance startup. |
| over allocation count | This statistic is cumulative since instance startup. Over allocating PGA memory can happen if the value of PGA_AGGREGATE_TARGET is too small. When this happens, the Oracle Database cannot honor the value of PGA_AGGREGATE_TARGET and extra PGA memory needs to be allocated.<br>If over allocation occurs, then increase the value of PGA_AGGREGATE_TARGET using the information provided by the V$PGA_TARGET_ADVICE view. |
| PGA memory freed back to OS | Number of bytes of PGA memory freed back to the operating system, cumulated since instance startup. |
| process count | Number of processes active within up to the last 3 seconds. |
| recompute count (total) | Number of times the instance bound, which is a cap on the maximum size of each active work area, has been recomputed since instance startup. Generally, the instance bound is recomputed in the background every 3 seconds, but it could be recomputed by a foreground process when the number of work areas changes rapidly in a short period of time. |
| total freeable PGA memory | Number of bytes of PGA memory in all processes that could be freed back to the operating system. |
| total PGA allocated | Current amount of PGA memory allocated by the instance. The Oracle Database attempts to keep this number below the value of the PGA_AGGREGATE_TARGET initialization parameter. However, it is possible for the PGA allocated to exceed that value by a small percentage and for a short period of time when the work area workload is increasing very rapidly or when PGA_AGGREGATE_TARGET is set to a small value. |
| total PGA inuse | Indicates how much PGA memory is currently consumed by work areas. This number can be used to determine how much memory is consumed by other consumers of the PGA memory (for example, PL/SQL or Java). |
| total PGA used for auto workareas | Indicates how much PGA memory is currently consumed by work areas running under the automatic memory management mode. This number can be used to determine how much memory is consumed by other consumers of the PGA memory (for example, PL/SQL or Java). |

| Statistic Name | Description |
|---|---|
| total PGA used for manual workareas | Indicates how much PGA memory is currently consumed by work areas running under the manual memory management mode. This number can be used to determine how much memory is consumed by other consumers of the PGA memory (for example, PL/SQL or Java). |

**See Also:**

"PGA_AGGREGATE_TARGET"

# V$PQ_SESSTAT

V$PQ_SESSTAT lists session statistics for parallel queries. After you have run a query or DML operation, you can use the information derived from V$PQ_SESSTAT to view the number of slave processes used, and other information for the session and system.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC | VARCHAR2(30) | Name of the statistic:<br>• Queries Parallelized - Number of queries run in parallel<br>• DDL Parallelized - Number of DDL operations run in parallel<br>• DML Parallelized - Number of DML operations run in parallel<br>• DFO Trees - Number of executed DFO trees<br>• DOP - Degree of parallelism used for the last statement<br>• Server Threads - Number of PX servers used<br>• Allocation Height - Requested number of servers per instance<br>• Allocation Width - Requested number of instances<br>• Local Msgs Sent - Number of local (intra-instance) messages sent<br>• Distr Msgs Sent - Number of remote (inter-instance) messages sent<br>• Local Msgs Recv'd - Number of local (intra-instance) messages received<br>• Distr Msgs Recv'd - Number of remote (inter-instance) messages received<br>• Slave Sets - Total number of slave sets used for the last statement. For multiple parallelizers in the parallel plan, it is the cumulative count. |
| LAST_QUERY | NUMBER | Value of the statistic for the last operation |
| SESSION_TOTAL | NUMBER | Value of the statistic for the entire session to this point in time.<br>The value of the statistic for the entire session to this point in time is not maintained for the DOP, Server Threads, Allocation Height, Allocation Width, and Slave Set statistics. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PQ_SLAVE

V$PQ_SLAVE lists statistics for each of the active parallel execution servers on an instance.

| Column | Datatype | Description |
|---|---|---|
| SLAVE_NAME | VARCHAR2(4) | Name of the parallel execution server |
| STATUS | VARCHAR2(4) | Current status of the parallel execution server:<br>• BUSY<br>• IDLE |
| SESSIONS | NUMBER | Number of sessions that have used this parallel execution server |
| IDLE_TIME_CUR | NUMBER | Amount of time spent idle while processing statements in the current session |
| BUSY_TIME_CUR | NUMBER | Amount of time spent busy while processing statements in the current session |
| CPU_SECS_CUR | NUMBER | Amount of CPU time spent on the current session |
| MSGS_SENT_CUR | NUMBER | Number of messages sent while processing statements for the current session |
| MSGS_RCVD_CUR | NUMBER | Number of messages received while processing statements for the current session |
| IDLE_TIME_TOTAL | NUMBER | Total amount of time this query server has been idle |
| BUSY_TIME_TOTAL | NUMBER | Total amount of time this query server has been active |
| CPU_SECS_TOTAL | NUMBER | Total amount of CPU time this query server has used to process statements |
| MSGS_SENT_TOTAL | NUMBER | Total number of messages this query server has sent |
| MSGS_RCVD_TOTAL | NUMBER | Total number of messages this query server has received |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PQ_SYSSTAT

V$PQ_SYSSTAT lists system statistics for parallel queries. After you have run a query or DML operation, you can use the information derived from V$PQ_SYSSTAT to view the number of slave processes used, and other information for the system.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC | VARCHAR2(30) | Name of the statistic:<br>• Servers Busy - Number of currently busy servers on this instance<br>• Servers Idle - Number of currently idle servers on this instance<br>• Servers Highwater - Number of active servers on this instance that have partaken in >= 1 operation so far<br>• Server Sessions - Total number of operations executed in all servers on this instance<br>• Servers Started - Total number of servers started on this instance<br>• Servers Shutdown - Total number of servers shutdown on this instance<br>• Servers Cleaned Up - Total number of servers on this instance cleaned up due to process death<br>• Queries Initiated - Total number of parallel queries initiated on this instance<br>• DDL Initiated - Total number of parallel DDL operations that were initiated<br>• DML Initiated - Total number of parallel DML operations that were initiated<br>• DFO Trees - Total number of DFO trees executed on this instance<br>• Local Msgs Sent - Total number of local (intra-instance) messages sent on this instance<br>• Distr Msgs Sent - Total number of remote (inter-instance) messages sent on this instance<br>• Local Msgs Recv'd - Total number of remote (inter-instance) messages received on this instance<br>• Distr Msgs Recv'd - Total number of remote (inter-instance) messages received on this instance |
| VALUE | NUMBER | Value of the statistic |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PQ_TQSTAT

V$PQ_TQSTAT contains statistics on parallel execution operations. The statistics are compiled after the query completes and only remain for the duration of the session. It displays the number of rows processed through each parallel execution server at each

stage of the execution tree. This view can help determine skew problems in a query's execution. (Note that for PDML, information from V$PQ_TQSTAT is available only after a commit or rollback operation.)

| Column | Datatype | Description |
|---|---|---|
| DFO_NUMBER | NUMBER | Data flow operator (DFO) tree number to differentiate queries |
| TQ_ID | NUMBER | Table queue ID within the query, which represents the connection between two DFO nodes in the query execution tree |
| SERVER_TYPE | VARCHAR2(10) | The role in table queue - producer/consumer/ranger |
| NUM_ROWS | NUMBER | The number of rows produced/consumed |
| BYTES | NUMBER | The number of bytes produced/consumed |
| OPEN_TIME | NUMBER | Time (seconds) the table queue remained open |
| AVG_LATENCY | NUMBER | Time (minutes) for a message to be dequeued after it enters the queue |
| WAITS | NUMBER | The number of waits encountered during dequeue |
| TIMEOUTS | NUMBER | The number of timeouts when waiting for a message |
| PROCESS | VARCHAR2(6) | Process ID |
| INSTANCE | NUMBER | Instance ID |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROCESS

V$PROCESS displays information about the currently active processes.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of the process state object |
| PID | NUMBER | Oracle process identifier |
| SOSID | VARCHAR2(24) | Operating system (process, thread) identifier.<br>This identifier is unique whether the Oracle multiprocess/multithread feature is enabled or not. |
| SPID | VARCHAR2(24) | Operating system process identifier.<br>The Oracle multiprocess/multithread feature is available for UNIX systems.<br>When the Oracle multiprocess/multithread feature is enabled, RDBMS processes are mapped to threads running in operating system processes, and the SPID identifier is not unique for RDBMS processes. |

| Column | Datatype | Description |
|---|---|---|
| | | When the Oracle multiprocess/multithread feature is not enabled on UNIX systems, the SPID identifier is unique for RDBMS processes. |
| STID | VARCHAR2(24) | Operating system thread identifier.<br>The Oracle multiprocess/multithread feature is available for UNIX systems.<br>When the Oracle multiprocess/multithread feature is enabled, RDBMS processes are mapped to threads running in operating system processes, and the SPID and STID together uniquely identify an RDBMS process.<br>The STID is not unique on Solaris. The STID is unique on Linux and AIX. |
| EXECUTION_TYPE | VARCHAR2(10) | Operating system execution type |
| PNAME | VARCHAR2(5) | Name of this process |
| USERNAME | VARCHAR2(15) | Operating system process username |
| SERIAL# | NUMBER | Process serial number |
| TERMINAL | VARCHAR2(30) | Operating system terminal identifier |
| PROGRAM | VARCHAR2(48) | Program in progress |
| TRACEID | VARCHAR2(255) | Trace file identifier |
| TRACEFILE | VARCHAR2(513) | Trace file name of the process |
| BACKGROUND | VARCHAR2(1) | 1 for a SYSTEM background process; NULL for foreground processes or non-SYSTEM background processes |
| LATCHWAIT | VARCHAR2(16) | Address of the latch the process is waiting for; NULL if none |
| LATCHSPIN | VARCHAR2(16) | This column is obsolete |
| PGA_USED_MEM | NUMBER | PGA memory currently used by the process (in bytes) |
| PGA_ALLOC_MEM | NUMBER | PGA memory currently allocated by the process (including free PGA memory not yet released to the operating system by the server process), in bytes |
| PGA_FREEABLE_MEM | NUMBER | Allocated PGA memory which can be freed (in bytes) |
| PGA_MAX_MEM | NUMBER | Maximum PGA memory ever allocated by the process (in bytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROCESS_MEMORY

V$PROCESS_MEMORY displays dynamic PGA memory usage by named component categories for each process.

| Column | Datatype | Description |
|---|---|---|
| PID | NUMBER | Oracle process identifier |
| SERIAL# | NUMBER | Oracle process serial number |
| CATEGORY | VARCHAR2(15) | Category name. Categories include "SQL", "PL/SQL", "OLAP" and "JAVA". Special categories are "Freeable" and "Other". Freeable memory has been allocated to the process by the operating system, but has not been allocated to a category. "Other" memory has been allocated to a category, but not to one of the named categories. |
| ALLOCATED | NUMBER | Bytes of PGA memory allocated by the process for the category. For the "Freeable" category, it is the amount of free PGA memory eligible to be released to the operating system. |
| USED | NUMBER | Bytes of PGA memory used by the process for the category. For "Freeable", the value is zero. For "Other", the value is NULL for performance reasons. |
| MAX_ALLOCATED | NUMBER | Maximum bytes of PGA memory ever allocated by the process for the category. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROPAGATION_RECEIVER

V$PROPAGATION_RECEIVER displays information about buffer queue propagation schedules on the receiving (destination) side. The values are reset to zero when the database (or instance in an Oracle Real Application Clusters (Oracle RAC) environment) restarts, when propagation migrates to another instance, or when an unscheduled propagation is attempted.

| Column | Datatype | Description |
|---|---|---|
| SRC_QUEUE_SCHEMA | VARCHAR2(30) | Name of the source schema |
| SRC_QUEUE_NAME | VARCHAR2(30) | Name of the source queue |
| SRC_DBNAME | VARCHAR2(128) | Name of the source database |
| DST_QUEUE_SCHEMA | VARCHAR2(30) | Name of the destination schema |
| DST_QUEUE_NAME | VARCHAR2(30) | Name of the destination queue |

| Column | Datatype | Description |
|---|---|---|
| STARTUP_TIME | DATE | Startup time of this schedule. This time changes when the source/destination database gets restarted. |
| HIGH_WATER_MARK | NUMBER | High watermark of the messages received |
| ACKNOWLEDGEMENT | NUMBER | Acknowledgement of the messages received by the receiver |
| LAST_RECEIVED_MSG | NUMBER | Last received message |
| TOTAL_MSGS | NUMBER | Total number of messages |
| ELAPSED_UNPICKLE_TIME | NUMBER | Elapsed unpickle time |
| ELAPSED_RULE_TIME | NUMBER | Elapsed rule time |
| ELAPSED_ENQUEUE_TIME | NUMBER | Elapsed enqueue time |
| SESSION_ID | NUMBER | Session ID of the propagation receiver |
| SERIAL# | NUMBER | Serial number of the propagation receiver |
| SPID | VARCHAR2(24) | Process identification number of the propagation receiver |
| PROPAGATION_NAME | VARCHAR2(30) | Name of the Streams propagation on the source database |
| STATE | VARCHAR2(43) | State of the propagation receiver:<br>• Initializing<br>• Sending unapplied txns<br>• Waiting for message from client<br>• Receiving LCRs<br>• Evaluating rules<br>• Enqueueing LCRS<br>• Waiting for memory<br>• Waiting for apply to read<br>• Waiting for message from propagation sender<br>When the propagation schedule is not optimized, the state is Normal. |
| LAST_RECEIVED_MSG_POSITION | RAW(64) | Last received message position. Corresponds to LAST_RECEIVED_MSG, except the value is in position rather than SCN. Position is used by XStream to determine ordering. |
| ACKNOWLEDGEMENT_POSITION | RAW(64) | Acknowledgement position of the messages received by the receiver. Corresponds to ACKNOWLEDGEMENT, except the value is in position rather than SCN. Position is used by XStream to determine ordering. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROPAGATION_SENDER

V$PROPAGATION_SENDER displays information about buffer queue propagation schedules on the sending (source) side. The values are reset to zero when the database (or instance in an Oracle Real Application Clusters (Oracle RAC) environment) restarts, when propagation migrates to another instance, or when an unscheduled propagation is attempted.

| Column | Datatype | Description |
|---|---|---|
| QUEUE_ID | NUMBER | Queue identifier of the queue |
| QUEUE_SCHEMA | VARCHAR2(30) | Schema of the queue |
| QUEUE_NAME | VARCHAR2(30) | Name of the queue |
| DST_QUEUE_SCHEMA | VARCHAR2(30) | Destination schema of the queue |
| DST_QUEUE_NAME | VARCHAR2(30) | Name of the destination queue |
| STARTUP_TIME | DATE | Time at which the propagation started |
| DBLINK | VARCHAR2(128) | Name of the destination database link |
| HIGH_WATER_MARK | NUMBER | High watermark of the messages sent |
| ACKNOWLEDGEMENT | NUMBER | Acknowledgement of the messages received by the receiver |
| SCHEDULE_STATUS | VARCHAR2(30) | Status of the propagation schedule |
| TOTAL_MSGS | NUMBER | Total messages propagated |
| TOTAL_BYTES | NUMBER | Total bytes propagated |
| ELAPSED_DEQUEUE_TIME | NUMBER | Elapsed dequeue time (in hundredths of a second) |
| ELAPSED_PICKLE_TIME | NUMBER | Elapsed pickle time (time taken to linearize a logical change record (LCR) into a stream of bytes that can be sent over the network) (in hundredths of a second) |
| ELAPSED_PROPAGATION_TIME | NUMBER | Elapsed propagation time (in hundredths of a second) |
| ELAPSED_RULE_TIME | NUMBER | Elapsed rule time (in hundredths of a second) |
| MAX_NUM_PER_WIN | NUMBER | Maximum bytes per window |
| MAX_SIZE | NUMBER | Maximum bytes sent per window |
| LAST_MSG_LATENCY | NUMBER | Last propagated message latency |
| LAST_MSG_ENQUEUE_TIME | TIMESTAMP(6) | Last propagated message enqueue time |
| LAST_MSG_PROPAGATION_TIME | TIMESTAMP(6) | Last time when the message was propagated |
| LAST_LCR_LATENCY | NUMBER | Last propagated LCR latency |

| Column | Datatype | Description |
|---|---|---|
| LAST_LCR_CREATION_TIME | DATE | Last propagated LCR timestamp |
| LAST_LCR_PROPAGATION_TIME | DATE | Last time when the LCR was propagated |
| DST_DATABASE_NAME | VARCHAR2(128) | Global name of the destination database |
| SESSION_ID | NUMBER | Session ID of the propagation sender process |
| SERIAL# | NUMBER | Serial number of the propagation sender process |
| SPID | VARCHAR2(24) | Process identification number of the propagation sender process |
| PROPAGATION_NAME | VARCHAR2(30) | Name of the Streams propagation |
| STATE | VARCHAR2(53) | State of the propagation sender process:<br>• Initializing<br>• Initializing propagation receiver<br>• Browsing LCRs<br>• Evaluating rules<br>• Dequeueing LCRs<br>• Sending LCRs<br>• Waiting for apply to be enabled<br>• Waiting for apply database to start<br>• Waiting for propagation to be enabled<br>• Waiting for capture to terminate<br>• Waiting for a subscriber to be added<br>• Suspended due to a dropped subscriber<br>• Suspended for auto split/merge<br>• Waiting on empty queue<br>When the SCHEDULE_STATUS column is not SCHEDULE OPTIMIZED, the state is the value of the SCHEDULE_STATUS column. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROXY_ARCHIVEDLOG

V$PROXY_ARCHIVEDLOG contains descriptions of archived log backups that were taken using the proxy copy functionality.

In a proxy copy, the media manager takes over the operations of backing up and restoring data. Each row represents a backup of one control file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Proxy copy record identifier |

| Column | Datatype | Description |
|---|---|---|
| STAMP | NUMBER | Proxy copy stamp |
| DEVICE_TYPE | VARCHAR2(17) | Type of media device that stores the proxy copy |
| HANDLE | VARCHAR2(513) | Name or "handle" for the proxy copy |
| COMMENTS | VARCHAR2(81) | Comments about the proxy copy |
| MEDIA | VARCHAR2(65) | A comment that contains further information about the media manager that created this backup |
| MEDIA_POOL | NUMBER | Number of the media pool in which the proxy copy is stored |
| TAG | VARCHAR2(32) | Tag for the proxy copy |
| STATUS | VARCHAR2(1) | Status of the backup set:<br>• A - Available<br>• U - Unavailable<br>• X - Expired<br>• D - Deleted |
| DELETED | VARCHAR2(3) | Indicates whether this record has been deleted (YES) or not (NO) |
| THREAD# | NUMBER | Number of the redo thread |
| SEQUENCE# | NUMBER | Log sequence number |
| RESETLOGS_CHANGE# | NUMBER | RESETLOGS SCN of the database incarnation to which this archived log belongs |
| RESETLOGS_TIME | DATE | RESETLOGS time stamp of the database incarnation to which this archived log belongs |
| FIRST_CHANGE# | NUMBER | First SCN of this redo log |
| FIRST_TIME | DATE | Time when Oracle switched into the redo log |
| NEXT_CHANGE# | NUMBER | First SCN of the next redo log in the thread |
| NEXT_TIME | DATE | First time stamp of the next redo log in the thread |
| BLOCKS | NUMBER | Size of this archived redo log (in operating system blocks) |
| BLOCK_SIZE | NUMBER | Block size for the copy (in bytes) |
| START_TIME | DATE | Time when the proxy copy was initiated |
| COMPLETION_TIME | DATE | Time when the proxy copy was completed |
| ELAPSED_SECONDS | NUMBER | Duration of the proxy copy |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS stamp |
| TERMINAL | VARCHAR2(3) | Indicates whether this record corresponds to a terminal archived redo log, as defined in V$ARCHIVED_LOG (YES) or not (NO) |

| Column | Datatype | Description |
|---|---|---|
| KEEP | VARCHAR2(3) | Indicates whether this backup set has a retention policy that is different than the value for the configure retention policy (YES) or not (NO) |
| KEEP_UNTIL | DATE | If specified, then this is the date after which the backup becomes obsolete. If this column is NULL, then the backup never expires. |
| KEEP_OPTIONS | VARCHAR2(11) | Additional retention options for this backup set:<br>• LOGS - Indicates a long-term backup made with the LOGS keyword, which is now deprecated<br>• BACKUP_LOGS - Indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent<br>• NOLOGS - Indicates a consistent backup made when the database was mounted<br>• NULL - Indicates that this backup has no KEEP options and becomes obsolete based on the retention policy |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROXY_ARCHIVELOG_DETAILS

V$PROXY_ARCHIVELOG_DETAILS contains information about all available archive log proxy copies.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session recid |
| SESSION_STAMP | NUMBER | Session stamp |
| COPY_KEY | NUMBER | Copy identifier |
| THREAD# | NUMBER | Redo thread number |
| SEQUENCE# | NUMBER | Redo log sequence number |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change number of the database when this log was written |
| RESETLOGS_TIME | DATE | Resetlogs time of the database when this log was written |
| HANDLE | VARCHAR2(513) | Proxy copy handle identifies the copy for restore |
| MEDIA | VARCHAR2(65) | Name of the media on which the copy resides. This value is informational only. It is not needed for restore. |

| Column | Datatype | Description |
|---|---|---|
| MEDIA_POOL | NUMBER | Media pool in which the copy resides. This is the same value that was entered in the POOL operand of the Recovery Manager BACKUP command. |
| TAG | VARCHAR2(32) | Proxy copy tag |
| FIRST_CHANGE# | NUMBER | First change number in the archived log |
| NEXT_CHANGE# | NUMBER | First change number in the next log |
| FIRST_TIME | DATE | Timestamp of the first change |
| NEXT_TIME | DATE | Timestamp of the next change |
| OUTPUT_BYTES | NUMBER | Total output bytes written |
| COMPLETION_TIME | DATE | Completion time |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(3) | Displayable format for output bytes |
| KEEP | DATE | Indicates whether this backup set has a retention policy that is different than the value for the configure retention policy (YES) or not (NO) |
| KEEP_UNTIL | VARCHAR2(11) | If specified, then this is the date after which the backup becomes obsolete. If this column is NULL, then the backup never expires. |
| KEEP_OPTIONS | VARCHAR2(4000) | Additional retention options for this backup set:<br>• LOGS - Indicates a long-term backup made with the LOGS keyword, which is now deprecated<br>• BACKUP_LOGS - Indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent<br>• NOLOGS - Indicates a consistent backup made when the database was mounted<br>• NULL - Indicates that this backup has no KEEP options and becomes obsolete based on the retention policy |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROXY_ARCHIVELOG_SUMMARY

V$PROXY_ARCHIVELOG_SUMMARY provides summary information about the output proxy archive log file.

| Column | Datatype | Description |
|---|---|---|
| NUM_FILES_BACKED | NUMBER | Number of archived log files backed up |

| Column | Datatype | Description |
|---|---|---|
| NUM_DISTINCT_FILES_BACKED | NUMBER | Number of distinct archived log files backed up |
| MIN_FIRST_CHANGE# | NUMBER | Minimum first change SCN |
| MAX_NEXT_CHANGE# | NUMBER | Maximum first change SCN |
| MIN_FIRST_TIME | DATE | Minimum first change time. Forms the redo range, along with MAX_NEXT_TIME. |
| MAX_NEXT_TIME | DATE | Maximum next change time |
| OUTPUT_BYTES | NUMBER | Total output size, in bytes |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROXY_COPY_DETAILS

V$PROXY_COPY_DETAILS contains information about all available control file and datafile proxy copies.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session recid |
| SESSION_STAMP | NUMBER | Session stamp |
| COPY_KEY | NUMBER | Copy identifier |
| FILE# | NUMBER | Absolute datafile number, or 0 if this is a control file backup |
| HANDLE | VARCHAR2(513) | Proxy copy handle identifies the copy for restore |
| MEDIA | VARCHAR2(65) | Name of the media on which the copy resides. This value is informational only. It is not needed for restore. |
| MEDIA_POOL | NUMBER | Media pool in which the copy resides. This is the same value that was entered in the POOL operand of the Recovery Manager BACKUP command. |
| TAG | VARCHAR2(32) | Proxy copy tag |
| CREATION_CHANGE# | NUMBER | Datafile creation change number |
| CREATION_TIME | DATE | Datafile creation timestamp |

| Column | Datatype | Description |
|---|---|---|
| CHECKPOINT_CHANGE# | NUMBER | Checkpoint change number of the datafile when the copy was made |
| CHECKPOINT_TIME | DATE | Checkpoint timestamp of the datafile when the copy was made |
| OUTPUT_BYTES | NUMBER | Total output bytes written |
| COMPLETION_TIME | DATE | Completion time |
| CONTROLFILE_TYPE | VARCHAR2(1) | Type of control file:<br>• B - Normal control file<br>• S - Standby control file |
| KEEP | VARCHAR2(3) | Indicates whether this backup set has a retention policy that is different than the value for the configure retention policy (YES) or not (NO) |
| KEEP_UNTIL | DATE | If specified, then this is the date after which the backup becomes obsolete. If this column is NULL, then the backup never expires. |
| KEEP_OPTIONS | VARCHAR2(11) | Additional retention options for this backup set:<br>• LOGS - Indicates a long-term backup made with the LOGS keyword, which is now deprecated<br>• BACKUP_LOGS - Indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent<br>• NOLOGS - Indicates a consistent backup made when the database was mounted.<br>• NULL - Indicates that this backup has no KEEP options and becomes obsolete based on the retention policy |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROXY_COPY_SUMMARY

V$PROXY_COPY_SUMMARY provides summary information about the output proxy datafile and control file.

| Column | Datatype | Description |
|---|---|---|
| NUM_COPIES | NUMBER | Number of copies created |
| NUM_DISTINCT_COPIES | NUMBER | Number of distinct copies (that contain datafiles with different checkpoints) |
| MIN_CHECKPOINT_CHANGE# | NUMBER | Minimum checkpoint change SCN |

| Column | Datatype | Description |
|---|---|---|
| MAX_CHECKPOINT_CHANGE# | NUMBER | Maximum checkpoint change SCN |
| MIN_CHECKPOINT_TIME | DATE | Minimum checkpoint change time |
| MAX_CHECKPOINT_TIME | DATE | Maximum checkpoint change time |
| OUTPUT_BYTES | NUMBER | Total output bytes |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Displayable format for output bytes |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PROXY_DATAFILE

V$PROXY_DATAFILE contains descriptions of datafile and control file backups that are taken with Proxy Copy. Each row represents a backup of one database file.

| Column | Datatype | Description |
|---|---|---|
| RECID | NUMBER | Proxy copy record ID |
| STAMP | NUMBER | Proxy copy record stamp |
| DEVICE_TYPE | VARCHAR2(17) | Type of the device on which the copy resides |
| HANDLE | VARCHAR2(513) | Proxy copy handle identifies the copy for restore |
| COMMENTS | VARCHAR2(81) | Comment returned by the operating system or storage subsystem. This value is informational only. It is not needed for restore. |
| MEDIA | VARCHAR2(65) | Name of the media on which the copy resides. This value is informational only. It is not needed for restore. |
| MEDIA_POOL | NUMBER | Media pool in which the copy resides. This is the same value that was entered in the POOL operand of the Recovery Manager BACKUP command |
| TAG | VARCHAR2(32) | Proxy copy tag |
| STATUS | VARCHAR2(1) | Status of the backup set:<br>• A - Available<br>• U - Unavailable<br>• X - Expired<br>• D - Deleted |
| DELETED | VARCHAR2(3) | Indicates whether this record has been deleted (YES) or not (NO) |

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Absolute datafile number, or 0 if this is a control file backup |
| CREATION_CHANGE# | NUMBER | Datafile creation change number |
| CREATION_TIME | DATE | Datafile creation Timestamp |
| RESETLOGS_CHANGE# | NUMBER | Resetlogs change number of the datafile when the copy was made |
| RESETLOGS_TIME | DATE | Resetlogs timestamp of the datafile when the copy was made |
| CHECKPOINT_CHANGE# | NUMBER | Checkpoint change number of the datafile when the copy was made |
| CHECKPOINT_TIME | DATE | Checkpoint timestamp of the datafile when the copy was made |
| ABSOLUTE_FUZZY_CHANGE# | NUMBER | Highest change in any block of the file, if known |
| RECOVERY_FUZZY_CHANGE# | NUMBER | Highest change written to the file by media recovery |
| RECOVERY_FUZZY_TIME | DATE | Timestamp of the highest change written to the file by media recovery |
| INCREMENTAL_LEVEL | NUMBER | If this backup is part of an incremental backup strategy, then 0. Otherwise null. |
| ONLINE_FUZZY | VARCHAR2(3) | Indicates whether this copy was made after a crash or offline immediate (or is a copy of a copy which was taken improperly while the database was open) (YES) or not (NO). Recovery will need to apply all redo up to the next crash recovery marker to make the file consistent. |
| BACKUP_FUZZY | VARCHAR2(3) | Indicates whether this is a copy taken using the BEGIN BACKUP \| END BACKUP technique (YES) or not (NO). The BEGIN BACKUP \| END BACKUP technique is used internally when proxy copies of open files are created. Recovery will need to apply all redo up to the end backup marker to make this copy consistent. |
| BLOCKS | NUMBER | Size of the copy (in blocks). Also the size of the datafile when the copy was made. |
| BLOCK_SIZE | NUMBER | Block size of the datafile |
| OLDEST_OFFLINE_RANGE | NUMBER | If the file number is 0 (that is, this is a control file backup), the RECID of the oldest offline range record in this control file copy. 0 for datafile copies. |
| START_TIME | DATE | Starting time |
| COMPLETION_TIME | DATE | Completion time |
| ELAPSED_SECONDS | NUMBER | Number of elapsed seconds |
| CONTROLFILE_TYPE | VARCHAR2(1) | Type of control file:<br>• B - Normal control file<br>• S - Standby control file |

| Column | Datatype | Description |
|---|---|---|
| KEEP | VARCHAR2(3) | Indicates whether this backup set has a retention policy that is different than the value for the configure retention policy (YES) or not (NO) |
| KEEP_UNTIL | DATE | If specified, then this is the date after which the backup becomes obsolete. If this column is NULL, then the backup never expires. |
| KEEP_OPTIONS | VARCHAR2(11) | Additional retention options for this backup set:<br>• LOGS - Indicates a long-term backup made with the LOGS keyword, which is now deprecated<br>• BACKUP_LOGS - Indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent<br>• NOLOGS - Indicates a consistent backup made when the database was mounted.<br>• NULL - Indicates that this backup has no KEEP options and becomes obsolete based on the retention policy |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS stamp |
| FOREIGN_DBID | NUMBER | Foreign DBID of the database from which this datafile was transported. The value is 0 if the file backed up is not a foreign database file. |
| PLUGGED_READONLY | VARCHAR2(3) | Indicates whether this is a proxy copy of a transported read-only foreign file (YES) or not (NO) |
| PLUGIN_CHANGE# | NUMBER | SCN at which the foreign datafile was transported into the database. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_CHANGE# | NUMBER | SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| PLUGIN_RESETLOGS_TIME | DATE | Time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| BACKED_BY_PDB | VARCHAR2(3) | Recovery Manager (RMAN) allows a PDB to be backed up in two ways. The value in this column indicates how the PDB backup was taken:<br>• YES: The backup was taken when connected to the PDB<br>• NO: The backup was taken when connected to the root container |

# V$PWFILE_USERS

V$PWFILE_USERS lists all users in the password file, and indicates whether the user has been granted the SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM privileges.

| Column | Datatype | Description |
|---|---|---|
| USERNAME | VARCHAR2(30) | Name of the user that is contained in the password file |
| SYSDBA | VARCHAR2(5) | Indicates whether the user can connect with SYSDBA privileges (TRUE) or not (FALSE) |
| SYSOPER | VARCHAR2(5) | Indicates whether the user can connect with SYSOPER privileges (TRUE) or not (FALSE) |
| SYSASM | VARCHAR2(5) | Indicates whether the user can connect with SYSASM privileges (TRUE) or not (FALSE) |
| SYSBACKUP | VARCHAR2(5) | Indicates whether the user can connect with SYSBACKUP privileges (TRUE) or not (FALSE) |
| SYSDG | VARCHAR2(5) | Indicates whether the user can connect with SYSDG privileges (TRUE) or not (FALSE) |
| SYSKM | VARCHAR2(5) | Indicates whether the user can connect with SYSKM privileges (TRUE) or not (FALSE) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PX_INSTANCE_GROUP

V$PX_INSTANCE_GROUP provides information about the instance groups being used for parallel operations by the current session.

| Column | Datatype | Description |
|---|---|---|
| QC_INSTANCE_GROUP | VARCHAR2(64) | The instance group being used by this session for parallel operations. This value repeats for every row returned. |
| WHY | VARCHAR2(23) | Where the current instance group name comes from, as follows:<br>SERVICE - the instance group being used is from the session's service name<br>PARALLEL_INSTANCE_GROUP - the instance group being used is being used because the PARALLEL_INSTANCE_GROUP initialization parameter has been set.<br>The value repeats for every row returned. |
| INSTANCE_NUMBER | NUMBER | Instance number of the instance providing this instance group. There will be one row for each instance in the instance group that the Query Coordinator is using. |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"PARALLEL_INSTANCE_GROUP"

# V$PX_PROCESS

V$PX_PROCESS contains information about the sessions running parallel execution.

| Column | Datatype | Description |
|---|---|---|
| SERVER_NAME | VARCHAR2(4) | The name of the PX server (P000, P001, and so on) |
| STATUS | VARCHAR2(9) | The state of the PX server (IN USE\|AVAILABLE) |
| PID | NUMBER | The process identifier |
| SPID | VARCHAR2(24) | Operating system process ID |
| SID | NUMBER | The session ID of the PX server, if in use |
| SERIAL# | NUMBER | The session serial number of the PX server, if in use |
| IS_GV | VARCHAR2(5) | Indicates whether a slave process in parallel is a normal one or a GV$ process.<br>Possible values:<br>• FALSE: The slave process in parallel is a normal process<br>• TRUE: The slave process in parallel is a GV$ process |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PX_PROCESS_SYSSTAT

V$PX_PROCESS_SYSSTAT contains information about the sessions running parallel execution.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC | VARCHAR2(30) | Name of the statistic: |

| Column | Datatype | Description |
|---|---|---|
| | | • `Servers In Use` - Number of PX servers currently performing parallel operations<br>• `Servers Available` - Number of PX servers available to perform parallel operations<br>• `Servers Started` - Number of times the system has had to create a PX server process<br>• `Servers Shutdown` - Number of times a PX server process has been shut down. A PX server process will be shut down if it has not been used recently.<br>If this value is large, then consider increasing the parameter. This will improve performance by avoiding the latency of PX server process creation.<br>• `Servers HWM` - Maximum number of concurrent PX server processes<br>If this number is equal to the `PARALLEL_MAX_SERVERS` initialization parameter, then consider increasing the parameter. This could allow you to increase your throughput, especially if your system is under-utilized and the `V $SYSSTAT` statistic "Parallel operations downgraded to serial" is large.<br>• `Servers Cleaned Up` - Number of times PMON had to clean up a PX server. This should only happen during abnormal termination of a parallel operation.<br>If this number is large, then you should determine the cause.<br>• Server Sessions - Total number of sessions created by all PX servers<br>• `Memory Chunks Allocated` - Number of large memory chunks allocated by PX servers<br>• `Memory Chunks Freed` - Number of large memory chunks freed<br>• `Memory Chunks Current` - Number of large memory chunks currently being used<br>• `Memory Chunks HWM` - Maximum number of concurrently allocated chunks<br>• `Buffers allocated` - Number of times a message buffer has been allocated<br>• `Buffers freed` - Number of times a message buffer has been freed<br>• `Buffers Current` - Number of message buffers currently being used<br>• `Buffers HWM` - Maximum number of concurrently allocated message buffers |
| `VALUE` | `NUMBER` | Value of the statistic |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PX_SESSION

V$PX_SESSION contains information about the sessions running parallel execution.

| Column | Datatype | Description |
|---|---|---|
| SADDR | RAW(4 \| 8) | Session address |
| SID | NUMBER | Session identifier |
| SERIAL# | NUMBER | Session serial number |
| QCSID | NUMBER | Session identifier of the parallel coordinator |
| QCSERIAL# | NUMBER | Session serial number of the parallel coordinator |
| QCINST_ID | NUMBER | Instance number on which the parallel coordinator is running |
| SERVER_GROUP | NUMBER | The logical group of servers to which this cluster database process belongs |
| SERVER_SET | NUMBER | The logical set of servers to which this cluster database process belongs. A single server group will have at most two server sets. |
| SERVER# | NUMBER | The logical number of the cluster database process within a server set |
| DEGREE | NUMBER | The degree of parallelism being used by the server set |
| REQ_DEGREE | NUMBER | The degree of parallelism that was requested by the user when the statement was issued and before any resource, multiuser, or load balancing reductions |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$PX_SESSTAT

V$PX_SESSTAT contains information about the sessions running parallel execution.

| Column | Datatype | Description |
|---|---|---|
| SADDR | RAW(4 \| 8) | Session address |
| SID | NUMBER | Session identifier |
| SERIAL# | NUMBER | Session serial number |
| QCSID | NUMBER | Session identifier of the parallel coordinator |
| QCSERIAL# | NUMBER | Session serial number of the parallel coordinator |
| QCINST_ID | NUMBER | Instance number on which the parallel coordinator is running |

| Column | Datatype | Description |
|---|---|---|
| SERVER_GROUP | NUMBER | The logical group of servers to which this cluster database process belongs |
| SERVER_SET | NUMBER | The logical set of servers that this cluster database process belongs to. A single server group will have at most two server sets. |
| SERVER# | NUMBER | The logical number of the cluster database process within a server set |
| DEGREE | NUMBER | The degree of parallelism being used by the server set |
| REQ_DEGREE | NUMBER | The degree of parallelism that was requested by the user when the statement was issued and before any resource, multiuser, or load balancing reductions |
| STATISTIC# | NUMBER | Statistic number |
| VALUE | NUMBER | Statistic value |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$QMON_COORDINATOR_STATS

V$QMON_COORDINATOR_STATS displays statistics of the non-sharded queue master process. There is one row per instance. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| QMNC_PID | VARCHAR2(24) | Non-sharded queue master process ID |
| STATUS | VARCHAR2(24) | Current status of the coordinator:<br>• DEAD<br>• INITIALIZING<br>• RUNNING TASK COORDINATOR<br>• WAITING<br>• ADDING SERVER |
| NUM_SERVERS | NUMBER | Number of QMON servers currently running |
| LAST_SERVER_START_TIME | TIMESTAMP(3) WITH TIME ZONE | Last server startup time |
| LAST_SERVER_PID | VARCHAR2(24) | Process ID of the last server process (Qnnn) created |
| NEXT_WAKEUP_TIME | TIMESTAMP(3) WITH TIME ZONE | Next wakeup time of the coordinator |

| Column | Datatype | Description |
|---|---|---|
| NEXT_READY_TIME | TIMESTAMP(3) WITH TIME ZONE | Ready time of the first delayed task |
| NEXT_EXPIRY_TIME | TIMESTAMP(3) WITH TIME ZONE | Expiry time of the next ready task |
| LAST_WAIT_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the coordinator went to sleep |
| LAST_FAILURE | VARCHAR2(32) | Last failure encountered |
| LAST_FAILURE_TIME | TIMESTAMP(3) WITH TIME ZONE | Last failure time |
| MAX_TASK_LATENCY | VARCHAR2(40) | Maximum task latency across all the servers (in seconds) |
| MIN_TASK_LATENCY | VARCHAR2(40) | Minimum task latency across all the servers (in seconds) |
| TOTAL_TASK_LATENCY | NUMBER | Cumulative latency across all the tasks (in seconds) |
| TOTAL_TASKS_EXECUTED | NUMBER | Cumulative number of tasks serviced by all the servers |
| MAX_SERVERS | NUMBER | Maximum number of servers present at any point of time |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

# V$QMON_SERVER_STATS

V$QMON_SERVER_STATS displays information and statistics about the active queue monitor server processes. There is one row per live queue monitor server process. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| QMNC_PID | VARCHAR2(24) | Non-sharded queue master process ID |
| SERVER_PID | VARCHAR2(24) | Process ID of the server |
| SERVER_NAME | VARCHAR2(48) | Name of the server |
| STATUS | VARCHAR2(40) | Current state of the server:<br>• UNUSED<br>• RUNNING<br>• IDLE WAIT<br>• EXITING<br>• NOT ACTIVE |
| SERVER_START_TIME | TIMESTAMP(3) WITH TIME ZONE | Start time of the server |

| Column | Datatype | Description |
|---|---|---|
| TASK_NAME | VARCHAR2(32) | Current executing task |
| TASK_NUMBER | NUMBER | Unique task number of the running task |
| TASK_START_TIME | TIMESTAMP(3) WITH TIME ZONE | Start time of the running task |
| LAST_WAIT_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the server last waited |
| MAX_LATENCY | NUMBER | Maximum task latency for this server (in seconds) |
| MIN_LATENCY | NUMBER | Minimum task latency for this server (in seconds) |
| TOTAL_LATENCY | NUMBER | Cumulative task latency for this server (in seconds) |
| NUM_TASKS | NUMBER | Number of tasks processed by the server |
| LAST_FAILURE | VARCHAR2(128) | Last failure encountered by the server |
| LAST_FAILURE_TIME | TIMESTAMP(3) WITH TIME ZONE | Last failure time |
| LAST_FAILURE_TASK | VARCHAR2(32) | Task being run at the time of the last failure |
| LAST_FAILURE_TASKNUM | NUMBER | Unique task number of the failed task |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

# V$QMON_TASK_STATS

V$QMON_TASK_STATS displays information and statistics based on different queue monitor tasks in the system (spilling, time manager activity, and so on). There is one row per kind of task. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| TASK_NAME | VARCHAR2(32) | Name of the task |
| TASK_TYPE | VARCHAR2(40) | Type of the task |
| LAST_CREATED_TASKNUM | NUMBER | Unique task number last created for this task |
| NUM_TASKS | NUMBER | Number of tasks currently present |
| TOTAL_TASK_RUN_TIME | NUMBER | Cumulative task run time |
| TOTAL_TASK_RUNS | NUMBER | Cumulative task runs |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_TASK_FAILURES | NUMBER | Cumulative failures |
| METRIC_TYPE | VARCHAR2(50) | Type of metric gathered for this task type |
| METRIC_VALUE | NUMBER | Value of this metric |
| LAST_FAILURE | VARCHAR2(32) | Last failure encountered while executing this type of task |
| LAST_FAILURE_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the last failure occurred |
| LAST_FAILURE_TASKNUM | NUMBER | Task number of the last failed task for this task |
| REMARK | VARCHAR2(64) | Remarks about the task |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

# V$QMON_TASKS

V$QMON_TASKS displays information and statistics about all queue background tasks in the system, which would be served by queue monitor servers. There is one row per task. The rows are deleted when the database (or instance in an Oracle RAC environment) restarts.

| Column | Datatype | Description |
|---|---|---|
| TASK_NAME | VARCHAR2(32) | Task name |
| TASK_NUMBER | NUMBER | Unique task number |
| TASK_TYPE | VARCHAR2(40) | Task type |
| TASK_SUBMIT_TIME | TIMESTAMP(3) WITH TIME ZONE | Task submit time |
| TASK_READY_TIME | TIMESTAMP(3) WITH TIME ZONE | Task ready time |
| TASK_EXPIRY_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when this task expires |
| TASK_START_TIME | TIMESTAMP(3) WITH TIME ZONE | Last actual start time for the task |
| TASK_STATUS | VARCHAR2(32) | Status of the task |
| SERVER_NAME | VARCHAR2(48) | Name of the QMON server running this task |
| MAX_RETRIES | NUMBER | Maximum retry count for the task |

| Column | Datatype | Description |
|---|---|---|
| NUM_RUNS | NUMBER | Number of runs of the task if repeatable |
| NUM_FAILURES | NUMBER | Number of failures encountered while running the task |
| CON_ID | NUMBER | The ID of the container to which the data pertains. The CON_ID value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

## V$QUEUE

V$QUEUE contains information on the shared server message queues.

| Column | Datatype | Description |
|---|---|---|
| PADDR | RAW(4 \| 8) | Address of the process that owns the queue |
| TYPE | VARCHAR2(10) | Type of queue:<br>• COMMON - Processed by servers<br>• DISPATCHER |
| QUEUED | NUMBER | Number of items in the queue |
| WAIT | NUMBER | Total time that all items in this queue have waited (in hundredths of a second). Divide by TOTALQ for average wait per item. |
| TOTALQ | NUMBER | Total number of items that have ever been in the queue |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$QUEUEING_MTH

V$QUEUEING_MTH displays all available queuing resource allocation methods.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(40) | Name of the queuing resource allocation method |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$RECOVER_FILE

V$RECOVER_FILE displays the status of files needing media recovery.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | File identifier number |
| ONLINE | VARCHAR2(7) | This column is obsolete and maintained for backward compatibility. The value of this column is always equal to the value in ONLINE_STATUS. |
| ONLINE_STATUS | VARCHAR2(7) | Online status (ONLINE, OFFLINE) |
| ERROR | VARCHAR2(18) | Why the file must be recovered: NULL if reason unknown, or OFFLINE NORMAL if recovery not needed |
| CHANGE# | NUMBER | SCN where recovery must start |
| TIME | DATE | Time of SCN when recovery must start |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RECOVERY_AREA_USAGE

V$RECOVERY_AREA_USAGE displays usage information about recovery areas.

| Column | Datatype | Description |
|---|---|---|
| FILE_TYPE | VARCHAR2(23) | File type:<br>• CONTROL FILE<br>• REDO LOG<br>• ARCHIVED LOG<br>• BACKUP PIECE<br>• IMAGE COPY<br>• FLASHBACK LOG<br>• REMOTE ARCHIVED LOG |
| PERCENT_SPACE_USED | NUMBER | Percent of the recovery area that is in use |
| PERCENT_SPACE_RECLAIMABLE | NUMBER | Percent of the recovery area that is reclaimable |
| NUMBER_OF_FILES | NUMBER | Number of files in the recovery area |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RECOVERY_FILE_DEST

V$RECOVERY_FILE_DEST displays information about the disk quota and current disk usage in the fast recovery area.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(513) | Location name. This is the value specified in the DB_RECOVERY_FILE_DEST initialization parameter. |
| SPACE_LIMIT | NUMBER | Maximum amount of disk space (in bytes) that the database can use for the fast recovery area. This is the value specified in the DB_RECOVERY_FILE_DEST_SIZE initialization parameter. |
| SPACE_USED | NUMBER | Amount of disk space (in bytes) used by fast recovery area files created in current and all previous fast recovery areas. Changing fast recovery areas does not reset SPACE_USED to 0. |
| SPACE_RECLAIMABLE | NUMBER | Total amount of disk space (in bytes) that can be created by deleting obsolete, redundant, and other low priority files from the fast recovery area |
| NUMBER_OF_FILES | NUMBER | Number of files in the fast recovery area |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "DB_RECOVERY_FILE_DEST"
- "DB_RECOVERY_FILE_DEST_SIZE"

# V$RECOVERY_FILE_STATUS

V$RECOVERY_FILE_STATUS contains one row for each datafile for each RECOVER statement. This view contains useful information only for the Oracle process doing the recovery. When Recovery Manager directs a server process to perform recovery, only Recovery Manager can view the relevant information in this view. V$RECOVERY_FILE_STATUS will be empty to all other Oracle users.

| Column | Datatype | Description |
|---|---|---|
| FILENUM | NUMBER | Number of the file being recovered |
| FILENAME | VARCHAR2(513) | Filename of the datafile being recovered |
| STATUS | VARCHAR2(13) | Status of the recovery:<br>• IN RECOVERY<br>• CURRENT |

| Column | Datatype | Description |
|---|---|---|
| | | • NOT RECOVERED |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Backup and Recovery User's Guide*

# V$RECOVERY_LOG

V$RECOVERY_LOG lists information about archived logs that are needed to complete media recovery. This information is derived from the log history view, V$LOG_HISTORY.

V$RECOVERY_LOG contains useful information only for the Oracle process doing the recovery. When Recovery Manager directs a server process to perform recovery, only Recovery Manager can view the relevant information in this view. V$RECOVERY_LOG will be empty to all other Oracle users.

| Column | Datatype | Description |
|---|---|---|
| THREAD# | NUMBER | Thread number of the archived log |
| SEQUENCE# | NUMBER | Sequence number of the archived log |
| TIME | DATE | Time of the first entry (lowest SCN) in the log |
| ARCHIVE_NAME | VARCHAR2(513) | Name of the file when archived, using the naming convention specified by the LOG_ARCHIVE_FORMAT initialization parameter<br>**See Also:** "LOG_ARCHIVE_FORMAT" |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$LOG_HISTORY" and *Oracle Database Backup and Recovery User's Guide*

# V$RECOVERY_PROGRESS

V$RECOVERY_PROGRESS can be used to track database recovery operations to ensure that they are not stalled, and also to estimate the time required to complete the operation in progress.

V$RECOVERY_PROGRESS is a subview of V$SESSION_LONGOPS.

| Column | Datatype | Description |
|---|---|---|
| START_TIME | DATE | Start time of the recovery operation |
| TYPE | VARCHAR2(64) | Type of recovery operation being performed:<br>• CRASH RECOVERY<br>• INSTANCE RECOVERY<br>• MEDIA RECOVERY |
| ITEM | VARCHAR2(32) | Item being measured.<br>When TYPE is CRASH RECOVERY or INSTANCE RECOVERY, the possible values are:<br>• Log Files<br>• Redo Blocks<br>When TYPE is MEDIA RECOVERY, the possible values are:<br>• Active Apply Rate<br>• Average Apply Rate<br>• Maximum Apply Rate<br>• Redo Applied<br>• Log Files<br>• Last Applied Redo<br>• Active Time<br>• Elapsed Time<br>• Apply Time per Log<br>• Checkpoint Time per Log<br>• Standby Apply Lag |
| UNITS | VARCHAR2(32) | The units of measurement for each item |
| SOFAR | NUMBER | Amount of work done so far |
| TOTAL | NUMBER | Total amount of work expected |
| TIMESTAMP | DATE | Timestamp of the last redo record applied |
| COMMENTS | VARCHAR2(176) | Miscellaneous notes; currently displays the SCN for the last applied redo |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SESSION_LONGOPS"
- *Oracle Database Backup and Recovery User's Guide* for more information about performing database recovery

# V$RECOVERY_STATUS

V$RECOVERY_STATUS contains statistics of the current recovery process. This view contains useful information only for the Oracle process doing the recovery. When Recovery Manager directs a server process to perform recovery, only Recovery Manager can view the relevant information in this view. V$RECOVERY_STATUS will be empty to all other Oracle users.

| Column | Datatype | Description |
|---|---|---|
| RECOVERY_CHECKPOINT | DATE | Point in time to which the recovery has occurred. If no logs have been applied, this is the point in time the recovery starts. |
| THREAD | NUMBER | Number of the redo thread currently being processed |
| SEQUENCE_NEEDED | NUMBER | Log sequence number of the log needed by the recovery process. The value is 0 if no log is needed. |
| SCN_NEEDED | VARCHAR2(16) | Low SCN of the log needed by recovery. The value is 0 if unknown or no log is needed. |
| TIME_NEEDED | DATE | Time when the log was created. The value is midnight on 1/1/88 if the time is unknown or if no log is needed. |
| PREVIOUS_LOG_NAME | VARCHAR2(513) | Filename of the log |
| PREVIOUS_LOG_STATUS | VARCHAR2(13) | Status of the previous log. Contains one of the following values: RELEASE; WRONG NAME; MISSING NAME; UNNEEDED NAME; NONE |
| REASON | VARCHAR2(13) | Reason recovery is returning control to the user (NEED LOG\|LOG REUSED\|THREAD DISABLED) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Backup and Recovery User's Guide*

# V$REDO_DEST_RESP_HISTOGRAM

V$REDO_DEST_RESP_HISTOGRAM provides statistical information for each redo transport destination.

| Column | Datatype | Description |
|---|---|---|
| DEST_ID | NUMBER | A nonnegative integer value representing a SYNC standby destination |
| TIME | VARCHAR2(20) | A text string that shows the last wall-clock time that a bucket was hit |
| DURATION | NUMBER | A positive integer value that represents a bucket of seconds, 1, 2, 3, up to 300 seconds, followed by 5 additional buckets that represent 600, 1200, 2400, 4800, and 9600 ( >= 4801) seconds |
| FREQUENCY | NUMBER | A nonnegative integer that shows the number of times a particular bucket was hit by the destination specified by DEST_ID |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$REPLPROP

V$REPLPROP displays information about the parallel propagation currently in progress at the replication site. Use this view to determine which transactions are currently being propagated, the number of calls propagated in each transaction, and the current activity of the parallel propagation coordinator process.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |
| SERIAL# | NUMBER | Session serial number. Used to identify uniquely a session's objects. Guarantees that session-level commands are applied to the correct session objects if the session ends with, and another session begins with, the same session ID. |
| NAME | VARCHAR2(71) | Name of the propagation session:<br>• Replication Parallel Prop Slave*n* - Slave process *n* is active, either waiting, pushing deferred transactions, purging metadata, or creating an error transaction<br>• Replication Parallel Prop Coordinator - Coordinator process is active, and either waiting, sleeping, or scheduling slaves to perform operations<br>The coordinator process reads transactions from the deferred transaction queue and assigns them to the slave processes. Then, the slaves propagate the transactions to the destination site. When the slaves push transactions in a push session, the slaves |

| Column | Datatype | Description |
|---|---|---|
| | | remain active until the push session completes, even if there are no more transactions to push. |
| DBLINK | VARCHAR2(128) | Database link on which this replication session is propagating |
| STATE | VARCHAR2(12) | State of the propagation session:<br>• WAIT - Slave processes or the coordinator process is waiting for an event (that is, a message)<br>• SLEEP - Coordinator process is sleeping for the duration of the delay_seconds setting.<br>**Note:** delay_seconds is set with the DBMS_DEFER_SYS.SCHEDULE_PUSH procedure.<br>• PUSH - Slave processes are pushing transactions from the deferred transaction queue to the remote site<br>• PURGE - Slave processes are purging metadata related to successfully applied transactions from the remote site<br>• CREATE ERROR - Slave processes are creating an error transaction. In this case, an error or a conflict occurred while the slaves were pushing deferred transactions to the remote site.<br>• SCHEDULE TXN - Coordinator process is determining the order that transactions are applied and assigning slave processes to execute the transactions |
| XID | VARCHAR2(22) | If the session is a slave session, then indicates the transaction id of the transaction that the slave is currently propagating. |
| SEQUENCE | NUMBER | If the process is a slave process, then indicates the sequence number of the calls propagated in the current operation, if relevant. Each transaction must process one or more calls, and the value of SEQUENCE starts at zero and increases as each call is processed. The SEQUENCE value shows the call that is currently being processed in each transaction. This value increases until the slave has processed all of the calls in a transaction. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view only contains data when deferred transactions are being pushed using parallel propagation at the current site. The parallelism parameter must be set to 1 or higher in the DBMS_DEFER_SYS.PUSH function for a push to use parallel propagation. Otherwise, the push uses serial propagation, and no data appears in this view during the push.

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_DEFER_SYS package

# V$REPLQUEUE

V$REPLQUEUE displays statistics about the replication deferred transactions queue. All values are stored since the start of the current database instance.

| Column | Datatype | Description |
|---|---|---|
| TXNS_ENQUEUED | NUMBER | Number of transactions enqueued in the deferred transactions queue |
| CALLS_ENQUEUED | NUMBER | Number of calls enqueued into the deferred transactions queue |
| TXNS_PURGED | NUMBER | Number of transactions purged from the deferred transactions queue |
| LAST_ENQUEUE_TIME | DATE | Date when the last transaction was enqueued into the deferred transaction queue (null if no transactions have been enqueued into the deferred transaction queue since the instance started) |
| LAST_PURGE_TIME | DATE | Date when the last transaction was purged from the deferred transaction queue (null if no transactions have been purged from the deferred transaction queue since the instance started) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$REQDIST

V$REQDIST lists statistics for the histogram of shared server dispatcher request times, divided into 12 buckets, or ranges of time. The time ranges grow exponentially as a function of the bucket number.

| Column | Datatype | Description |
|---|---|---|
| BUCKET | NUMBER | Bucket number: 0 - 11; the maximum time for each bucket is (4 * 2^N)/100 seconds |
| COUNT | NUMBER | Count of requests whose total time to complete (excluding wait time) falls in this range |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RESERVED_WORDS

V$RESERVED_WORDS displays a list of all SQL keywords. To determine whether a particular keyword is reserved in any way, check the RESERVED, RES_TYPE, RES_ATTR, and RES_SEMI columns.

| Column | Datatype | Description |
|---|---|---|
| KEYWORD | VARCHAR2(30) | Name of the keyword |
| LENGTH | NUMBER | Length of the keyword |
| RESERVED | VARCHAR2(1) | Indicates whether the keyword cannot be used as an identifier (Y) or whether the keyword is not reserved (N) |
| RES_TYPE | VARCHAR2(1) | Indicates whether the keyword cannot be used as a type name (Y) or whether the keyword is not reserved (N) |
| RES_ATTR | VARCHAR2(1) | Indicates whether the keyword cannot be used as an attribute name (Y) or whether the keyword is not reserved (N) |
| RES_SEMI | VARCHAR2(1) | Indicates whether the keyword is not allowed as an identifier in certain situations, such as in DML (Y) or whether the keyword is not reserved (N) |
| DUPLICATE | VARCHAR2(1) | Indicates whether the keyword is a duplicate of another keyword (Y) or whether the keyword is not a duplicate (N) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RESOURCE

V$RESOURCE contains resource name and address information.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of the resource object |
| TYPE | VARCHAR2(2) | Resource type; the resource types are listed in Table 8-1 |
| ID1 | NUMBER | Resource identifier #1 |
| ID2 | NUMBER | Resource identifier #2 |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RESOURCE_LIMIT

V$RESOURCE_LIMIT displays information about global resource use for some of the system resources. Use this view to monitor the consumption of resources so that you can take corrective action, if necessary. Many of the resources correspond to initialization parameters listed in Table 8-5.

Some resources, those used by DLM for example, have an initial allocation (soft limit), and the hard limit, which is theoretically infinite (although in practice it is limited by SGA size). During SGA reservation/initialization, a place is reserved in SGA for the INITIAL_ALLOCATION of resources, but if this allocation is exceeded, additional resources are allocated up to the value indicated by LIMIT_VALUE. The CURRENT_UTILIZATION column indicates whether the initial allocation has been exceeded. When the initial allocation value is exceeded, the additional required resources are allocated from the shared pool, where they must compete for space with other resources.

A good choice for the value of INITIAL_ALLOCATION will avoid the contention for space. For most resources, the value for INITIAL_ALLOCATION is the same as the LIMIT_VALUE. Exceeding LIMIT_VALUE results in an error.

| Column | Datatype | Description |
|---|---|---|
| RESOURCE_NAME | VARCHAR2(30) | Name of the resource (see Table 8-5) |
| CURRENT_UTILIZATION | NUMBER | Number of (resources, locks, or processes) currently being used |
| MAX_UTILIZATION | NUMBER | Maximum consumption of this resource since the last instance start-up |
| INITIAL_ALLOCATION | VARCHAR2(10) | Initial allocation. This will be equal to the value specified for the resource in the initialization parameter file (UNLIMITED for infinite allocation). |
| LIMIT_VALUE | VARCHAR2(10) | Unlimited for resources and locks. This can be greater than the initial allocation value (UNLIMITED for infinite limit). |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

*Table 8-5 Values for the RESOURCE_NAME Column*

| Resource Name | Corresponds to |
|---|---|
| DML_LOCKS | See "DML_LOCKS" |
| ENQUEUE_LOCKS | This value is computed by the Oracle Database. See V$ENQUEUE_LOCK to obtain more information about the enqueue locks. |
| GES_LOCKS | Global Enqueue Service locks |
| GES_PROCS | Global Enqueue Service processes |
| GES_RESS | Global Enqueue Service resources |
| MAX_SHARED_SERVERS | See "MAX_SHARED_SERVERS" |
| PARALLEL_MAX_SERVERS | See "PARALLEL_MAX_SERVERS" |
| PROCESSES | See "PROCESSES" |
| SESSIONS | See "SESSIONS" |
| SORT_SEGMENT_LOCKS | This value is computed by the Oracle Database |
| TEMPORARY_LOCKS | This value is computed by the Oracle Database |
| TRANSACTIONS | See "TRANSACTIONS" |

# V$RESTORE_POINT

V$RESTORE_POINT displays information about restore points.

| Column | Datatype | Description |
|---|---|---|
| SCN | NUMBER | Database SCN when the restore point was created |
| DATABASE_INCARNATION# | NUMBER | Database incarnation number when the restore point was created |
| GUARANTEE_FLASHBACK_DATABASE | VARCHAR2(3) | Indicates whether flashback log files will be kept to ensure a flashback to this point (YES) or not (NO) |
| STORAGE_SIZE | NUMBER | Approximate number of bytes of disk space currently tied up supporting this restore point. This will only be nonzero for guaranteed restore points. |
| TIME | TIMESTAMP(9) | Wall-clock time when the restore point was created |
| RESTORE_POINT_TIME | TIMESTAMP(9) | Time that was specified when the restore point was created. If a time was not specified, this value is NULL. |
| PRESERVED | VARCHAR2(3) | Indicates whether the restore point must be explicitly deleted (YES) or not (NO) |
| NAME | VARCHAR2(128) | Name of the restore point |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RESULT_CACHE_DEPENDENCY

V$RESULT_CACHE_DEPENDENCY displays the depends-on relationship between cached results and dependencies.

| Column | Datatype | Description |
|---|---|---|
| RESULT_ID | NUMBER | Cached result |
| DEPEND_ID | NUMBER | Dependency object |
| OBJECT_NO | NUMBER | Dictionary object number of the dependency object |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RESULT_CACHE_MEMORY

V$RESULT_CACHE_MEMORY displays all the memory blocks and their status.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Unique block identifier (that is, the block number) |
| CHUNK | NUMBER | Chunk to which the block belongs (the upper 27 bits of the ID) |
| OFFSET | NUMBER | Offset of the block within its chunk (the lower 5 bits of the ID) |
| FREE | VARCHAR2(3) | Indicates whether the block is free (YES) or not (NO) |
| OBJECT_ID | NUMBER | Cache object to which the memory block belongs; NULL if the memory block is not allocated to a cache object (FREE = YES) |
| POSITION | NUMBER | Position of the block in the cached object; NULL if the memory block is not allocated to a cache object (FREE = YES) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RESULT_CACHE_OBJECTS

V$RESULT_CACHE_OBJECTS displays all the objects (both cached results and dependencies) and their attributes.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Identifier for the cache object (also the ID of the first block) |
| TYPE | VARCHAR2(10) | Type of the cache object:<br>• Result<br>• Dependency |
| STATUS | VARCHAR2(9) | Status of the object:<br>• New - Result is still under construction<br>• Published - Result is available for use<br>• Bypass - Result will be bypassed from use<br>• Expired - Result has exceeded expiration time<br>• Invalid - Result is no longer available for use |
| BUCKET_NO | NUMBER | Internal hash bucket for the object |
| HASH | NUMBER | Hash value for the object |
| NAME | VARCHAR2(128) | Name (for example, SQL prefix or PL/SQL function name) |
| NAMESPACE | VARCHAR2(5) | Namespace:<br>• SQL<br>• PLSQL<br>• KEY VECTOR |
| CREATION_TIMESTAMP | DATE | Time when the object was created |
| CREATOR_UID | NUMBER | UID that created the object |
| DEPEND_COUNT | NUMBER | Number of dependencies (TYPE = Result) or dependents (TYPE = Dependency) |
| BLOCK_COUNT | NUMBER | Total number of blocks in the cached object |
| SCN | NUMBER | Build SCN (TYPE = Result) or invalidation SCN (TYPE = Dependency) |
| COLUMN_COUNT | NUMBER | Number of columns in the cached result |
| PIN_COUNT | NUMBER | Number of active scans on this result[1] |
| SCAN_COUNT | NUMBER | Total number of scans initiated on the cached result[1] |
| ROW_COUNT | NUMBER | Total number of rows in the cached result[1] |
| ROW_SIZE_MAX | NUMBER | Size of the largest row (in bytes)[1] |
| ROW_SIZE_MIN | NUMBER | Size of the smallest row (in bytes)[1] |
| ROW_SIZE_AVG | NUMBER | Average size of a row (in bytes)[1] |
| BUILD_TIME | NUMBER | Amount of time (in hundredths of a second) it took to build the cached result[1] |

| Column | Datatype | Description |
|---|---|---|
| LRU_NUMBER | NUMBER | LRU list position (the smaller the value, the more recent the usage)[1] |
| OBJECT_NO | NUMBER | Dictionary object number of the dependency object |
| INVALIDATIONS | NUMBER | Number of times the object has invalidated its dependents[2] |
| SPACE_OVERHEAD | NUMBER | Overhead (in bytes) for the result[1] |
| SPACE_UNUSED | NUMBER | Unused space (in bytes) for the result[1] |
| CACHE_ID | VARCHAR2(93) | CacheId for the result (object name if it's a dependency) |
| CACHE_KEY | VARCHAR2(93) | CacheKey for the result (object name if it's a dependency) |
| CHECKSUM | NUMBER | Checksum for the result object. The checksum is computed over all the blocks in the result cache object minus the object header. |
| EDITION_ID | NUMBER | Shows the edition's object ID that was in use when the result was calculated |
| DB_LINK[3] | VARCHAR2(3) | Possible values:<br>• YES: If the result cache object references a remote database object<br>• NO: If the result cache object does not reference a remote database object |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

1 These columns are only valid for TYPE = Result; otherwise, they are NULL.

2 These columns are only valid for TYPE = Dependency; otherwise, they are NULL.

3 This column is available starting in Oracle Database 12*c* Release 1 (12.1.0.2).

# V$RESULT_CACHE_STATISTICS

V$RESULT_CACHE_STATISTICS displays various Result Cache settings and usage statistics.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Statistic number |
| NAME | VARCHAR2(128) | Name of the statistic (see Table 8-6) |
| VALUE | VARCHAR2(81) | Value of the statistic |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

***Table 8-6 V$RESULT_CACHE_STATISTICS Statistics***

| Statistic Name | Description |
|---|---|
| Block Size (Bytes) | Size of each memory block |
| Block Count Maximum | Maximum number of memory blocks allowed |
| Block Count Current | Number of memory blocks currently allocated |
| Result Size Maximum (Blocks) | Maximum number of blocks allowed for a single result |
| Create Count Success | Number of cache results successfully created |
| Create Count Failure | Number of cache results that failed to create |
| Find Count | Number of cached results that were successfully found |
| Invalidation Count | Total number of invalidations |
| Delete Count Invalid | Number of invalid cached results deleted |
| Delete Count Valid | Number of valid cached results deleted |
| Hash Chain Length | Average length of items in the hash chain |
| Find Copy Count | Number of results copied directly out of the cache |

# V$RMAN_BACKUP_JOB_DETAILS

V$RMAN_BACKUP_JOB_DETAILS displays details about backup jobs.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Together, with SESSION_KEY and SESSION_STAMP, used to uniquely identify job output from V$RMAN_OUTPUT |
| SESSION_STAMP | NUMBER | Together, with SESSION_KEY and SESSION_RECID, used to uniquely identify job output from V$RMAN_OUTPUT |
| COMMAND_ID | VARCHAR2(33) | Either a user-specified SET COMMAND ID or a unique command ID generated by RMAN |
| START_TIME | DATE | Start time of the first BACKUP command in the job |
| END_TIME | DATE | End time of the last BACKUP command in the job |
| INPUT_BYTES | NUMBER | Sum of all input file sizes backed up by this job |
| OUTPUT_BYTES | NUMBER | Output size of all pieces generated by this job |
| STATUS_WEIGHT | NUMBER | Used internally by Enterprise Manager |

| Column | Datatype | Description |
|---|---|---|
| OPTIMIZED_WEIGHT | NUMBER | Used internally by Enterprise Manager |
| OBJECT_TYPE_WEIGHT | NUMBER | Used internally by Enterprise Manager |
| OUTPUT_DEVICE_TYPE | VARCHAR2(17) | Can be DISK, SBT, or *. An * indicates more than one device (in most cases, it will be DISK or SBT). |
| AUTOBACKUP_COUNT | NUMBER | Number of autobackups performed by this job |
| BACKED_BY_OSB | VARCHAR2(3) | A value of YES means the backup was done to Oracle Secure Backup. Otherwise, backed up by other third party tape library. |
| AUTOBACKUP_DONE | VARCHAR2(3) | YES or NO, depending upon whether or not a control file autobackup was done as part of this backup job |
| STATUS | VARCHAR2(23) | One of the following values:<br>• RUNNING WITH WARNINGS<br>• RUNNING WITH ERRORS<br>• COMPLETED<br>• COMPLETED WITH WARNINGS<br>• COMPLETED WITH ERRORS<br>• FAILED |
| INPUT_TYPE | VARCHAR2(13) | Contains one of the following values. If the user command does not satisfy one of them, then preference is given in order, from top to bottom of the list.<br>• DB FULL<br>• RECVR AREA<br>• DB INCR<br>• DATAFILE FULL<br>• DATAFILE INCR<br>• ARCHIVELOG<br>• CONTROLFILE<br>• SPFILE |
| OPTIMIZED | VARCHAR2(3) | YES or NO, depending on whether optimization was applied. Applicable to backup jobs only. |
| ELAPSED_SECONDS | NUMBER | Number of elapsed seconds |
| COMPRESSION_RATIO | NUMBER | Compression ratio |
| INPUT_BYTES_PER_SEC | NUMBER | Input read-rate-per-second |
| OUTPUT_BYTES_PER_SEC | NUMBER | Output write-rate-per-second |
| INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Values in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on. |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Values in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on |
| INPUT_BYTES_PER_SEC_DISPLAY | VARCHAR2(4000) | Input read-rate-per-second. These values are in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on. |

| Column | Datatype | Description |
|---|---|---|
| OUTPUT_BYTES_PER_SEC_DISPLAY | VARCHAR2(4000) | Output write-rate-per-second. These values are in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on. |
| TIME_TAKEN_DISPLAY | VARCHAR2(4000) | Time taken, shown in user-displayable format <nn>h:<nn>m:<nn>s |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RMAN_BACKUP_SUBJOB_DETAILS

V$RMAN_BACKUP_SUBJOB_DETAILS merges similar operations within an RMAN session into a single row. For example, if there are four BACKUP DATAFILE <n> commands, three RECOVERY COPY OF DATAFILE commands, and one BACKUP RECOVERY AREA command, this view will contain three rows - one each for BACKUP, ROLLFORWARD, and COPY_DISK_TO_TAPE operation.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Together with SESSION_KEY and SESSION_STAMP, used to uniquely identify job output from V$RMAN_OUTPUT |
| SESSION_STAMP | NUMBER | Together with SESSION_KEY and SESSION_RECID, used to uniquely identify job output from V$RMAN_OUTPUT |
| OPERATION | VARCHAR2(33) | Can be BACKUP, ROLLFORWARD, VALIDATE, or COPY_DISK_TO_TAPE. A row for each suboperation type for the session will be in the output view. |
| COMMAND_ID | VARCHAR2(33) | Either a user-specified SET COMMAND ID or a unique command ID generated by RMAN |
| START_TIME | DATE | Start time of the first BACKUP command in the job |
| END_TIME | DATE | End time of the last BACKUP command in the job |
| INPUT_BYTES | NUMBER | Sum of all input file sizes backed up by this job |
| OUTPUT_BYTES | NUMBER | Output size of all pieces generated by this job |
| STATUS_WEIGHT | NUMBER | Used internally by Enterprise Manager |
| OBJECT_TYPE_WEIGHT | NUMBER | Used internally by Enterprise Manager |
| OPTIMIZED_WEIGHT | NUMBER | Used internally by Enterprise Manager |
| OUTPUT_DEVICE_TYPE | VARCHAR2(17) | Can be DISK, SBT, or *. An * indicates more than one device (in most cases, it will be DISK or SBT). |

| Column | Datatype | Description |
|---|---|---|
| BACKED_BY_OSB | VARCHAR2(3) | A value of YES means the backup was done to Oracle Secure Backup. Otherwise, backed up by other third party tape library. |
| AUTOBACKUP_DONE | VARCHAR2(3) | YES or NO, depending upon whether or not a control file autobackup was done as part of this job |
| STATUS | VARCHAR2(23) | One of the following values:<br>• RUNNING WITH WARNINGS<br>• RUNNING WITH ERRORS<br>• COMPLETED<br>• COMPLETED WITH WARNINGS<br>• COMPLETED WITH ERRORS<br>• FAILED |
| INPUT_TYPE | VARCHAR2(13) | Contains one of the following values. If the user command does not satisfy one of them, then preference is given in order, from top to bottom of the list.<br>• DB FULL<br>• RECVR AREA<br>• DB INCR<br>• DATAFILE FULL<br>• DATAFILE INCR<br>• ARCHIVELOG<br>• CONTROLFILE<br>• SPFILE |
| OPTIMIZED | VARCHAR2(3) | YES or NO, depending on whether optimization was applied. Applicable to backup jobs only. |
| AUTOBACKUP_COUNT | NUMBER | Number of autobackups performed by this job |
| COMPRESSION_RATIO | NUMBER | Compression ratio |
| INPUT_BYTES_DISPLAY | VARCHAR2(4000) | Values in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on. |
| OUTPUT_BYTES_DISPLAY | VARCHAR2(4000) | Values in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RMAN_BACKUP_TYPE

V$RMAN_BACKUP_TYPE displays information about RMAN backup types.

| Column | Datatype | Description |
|---|---|---|
| WEIGHT | NUMBER | Used to set precedence order of different backup types in reports. |

| Column | Datatype | Description |
|---|---|---|
| INPUT_TYPE | VARCHAR2(13) | Used to represent possible filters used in creating various reporting screens. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RMAN_COMPRESSION_ALGORITHM

V$RMAN_COMPRESSION_ALGORITHM provides descriptions of supported compression algorithms. It is used by the RMAN client.

| Column | Datatype | Description |
|---|---|---|
| ALGORITHM_ID | NUMBER | Algorithm ID |
| ALGORITHM_NAME | VARCHAR2(64) | Name of the algorithm (for example, LOW, MEDIUM, DEFAULT, or HIGH) |
| INITIAL_RELEASE | VARCHAR2(18) | First Oracle Database release when this compression algorithm was available |
| TERMINAL_RELEASE | VARCHAR2(18) | Last Oracle Database release that supported using this compression algorithm to create new backups. Existing backups can always be restored, even if they use a deprecated compression algorithm. |
| ALGORITHM_DESCRIPTION | VARCHAR2(64) | Description of the algorithm |
| ALGORITHM_COMPATIBILITY | VARCHAR2(18) | Required database compatibility level for the algorithm (for example, 11.2.0 for DEFAULT) |
| IS_VALID | VARCHAR2(3) | Indicates whether the algorithm is valid with regard to the compatibility setting (YES) or not (NO). The value is YES if ALGORITHM_COMPATIBILITY <= DATABASE_COMPATIBILITY. |
| REQUIRES_ACO | VARCHAR2(3) | Indicates whether the algorithm requires the Advanced Compression Option (YES) or not (NO) |
| IS_DEFAULT | VARCHAR2(3) | Indicates whether the algorithm is the default compression algorithm that RMAN uses to create compressed backup sets (YES) or not (NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RMAN_CONFIGURATION

V$RMAN_CONFIGURATION lists information about RMAN persistent configuration settings.

| Column | Datatype | Description |
|---|---|---|
| CONF# | NUMBER | A unique key identifying this configuration record within the target database that owns it. |
| NAME | VARCHAR2(65) | The type of configuration. All options of the CONFIGURE command are valid types except:<br>• CONFIGURE EXCLUDE (described in RC_TABLESPACE)<br>• CONFIGURE AUXNAME (described in RC_DATAFILE)<br>• CONFIGURE SNAPSHOT CONTROLFILE (stored only in control file) |
| VALUE | VARCHAR2(1025) | The CONFIGURE command setting. Example: RETENTION POLICY TO RECOVERY WINDOW OF 10 DAYS |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RMAN_ENCRYPTION_ALGORITHMS

V$RMAN_ENCRYPTION_ALGORITHMS displays supported encryption algorithms. It is used by the RMAN client to validate user-requested algorithms. This view will list AES128, AES192, and AES256 encryption algorithms for the current release. The default algorithm is AES128.

| Column | Datatype | Description |
|---|---|---|
| ALGORITHM_ID | NUMBER | Number to identify the algorithm |
| ALGORITHM_NAME | VARCHAR2(64) | Name of the algorithm (for example, AES128, AES192, or AES256) |
| ALGORITHM_DESCRIPTION | VARCHAR2(64) | Description of the algorithm |
| IS_DEFAULT | VARCHAR2(3) | Indicates whether this is the default encryption algorithm (YES) or not (NO). This value is set by Oracle Database and may vary for each Release (that is, it is not dependent on user-specified RMAN configuration). |
| RESTORE_ONLY | VARCHAR2(3) | Indicates whether this algorithm can be used for restore only (YES) or not (NO). If the value is NO, then the algorithm is also allowed for backup. This column is useful in determining whether an encryption algorithm is deprecated for backup purpose. |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RMAN_OUTPUT

V$RMAN_OUTPUT displays messages reported by RMAN. This is an in-memory view and is not recorded in the controlfile. The view can hold 32768 rows.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the session which is running this RMAN operation |
| RECID | NUMBER | Record ID of the corresponding V$RMAN_STATUS row |
| STAMP | NUMBER | Timestamp of the corresponding V$RMAN_STATUS row |
| SESSION_RECID | NUMBER | Record ID of the session (corresponding V$RMAN_STATUS row with ROW_LEVEL = 0) |
| SESSION_STAMP | NUMBER | Timestamp of the session (corresponding V$RMAN_STATUS row with ROW_LEVEL = 0) |
| OUTPUT | VARCHAR2(130) | Output text reported by RMAN |
| RMAN_STATUS_RECID | NUMBER | Owning V$RMAN_STATUS record ID |
| RMAN_STATUS_STAMP | NUMBER | Owning V$RMAN_STATUS record stamp |
| SESSION_KEY | NUMBER | Session identifier |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RMAN_STATUS

V$RMAN_STATUS displays the finished and on-going RMAN jobs. For on-going jobs, this view displays progress and status. The jobs which are in progress are stored only in memory while the finished jobs are stored in the controlfile.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the session which is running this RMAN operation |
| RECID | NUMBER | Record ID of the row in the controlfile |

| Column | Datatype | Description |
|---|---|---|
| STAMP | NUMBER | Timestamp of the row (RECID + STAMP is unique) |
| PARENT_RECID | NUMBER | Record ID of the parent row of this row (corresponding V$RMAN_STATUS row with ROW_LEVEL = ROW_LEVEL - 1) |
| PARENT_STAMP | NUMBER | Timestamp of the parent row of this row (corresponding V$RMAN_STATUS row with ROW_LEVEL = ROW_LEVEL - 1) |
| SESSION_RECID | NUMBER | Record ID of the session (corresponding V$RMAN_STATUS row with ROW_LEVEL = 0) |
| SESSION_STAMP | NUMBER | Timestamp of the session (corresponding V$RMAN_STATUS row with ROW_LEVEL = 0) |
| ROW_LEVEL | NUMBER | Level of the row. The session has level 0. |
| ROW_TYPE | VARCHAR2(19) | Type of the row:<br>• SESSION<br>• COMMAND<br>• RECURSIVE OPERATION |
| COMMAND_ID | VARCHAR2(33) | Command ID set by the RMAN SET COMMAND ID command. If not set, then RMAN will create a unique number. |
| OPERATION | VARCHAR2(33) | Name of the command in the execution explained by this row |
| STATUS | VARCHAR2(23) | Status of the operation:<br>• RUNNING<br>• RUNNING WITH WARNINGS<br>• RUNNING WITH ERRORS<br>• COMPLETED<br>• COMPLETED WITH WARNINGS<br>• COMPLETED WITH ERRORS<br>• FAILED |
| MBYTES_PROCESSED | NUMBER | Percentage of the job completed; null if not applicable for the operation |
| START_TIME | DATE | Start time of the job |
| END_TIME | DATE | End time of the job |
| INPUT_BYTES | NUMBER | Number of input bytes read |
| OUTPUT_BYTES | NUMBER | Number of output bytes written |
| OPTIMIZED | VARCHAR2(3) | YES, if backup optimization was applied during the backup job. Otherwise, NO. |
| OBJECT_TYPE | VARCHAR2(13) | Identifies types of objects backed up |
| OUTPUT_DEVICE_TYPE | VARCHAR2(17) | DISK, SBT_TAPE, or *. An * indicates that output was written to more than one device type. |
| OSB_ALLOCATED | VARCHAR2(3) | A value of YES means an Oracle Secure Backup channel was allocated during the specified operation identified by the V$RMAN_STATUS view. |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RO_USER_ACCOUNT

V$RO_USER_ACCOUNT is populated only on Oracle databases that are open in read-only mode. When a database is read-only, security data cannot be stored in normal catalogue tables. Instead, the security data is stored in an in-memory table that is queried through this view.

In an Oracle Data Guard environment, some of the security information for user accounts on the standby is inherited from the primary server. For example, if the account is locked out unlimited on the primary, then it will be locked on the standby database(s). The information stored on the standby is volatile information that user actions on the standby database(s) can affect, such as the number of failed logins, and the time the account was locked on the standby due to failed access attempts. Note that failed login attempts on standbys do not affect the account status on primaries.

If this view is queried from the root in a multitenent container database (CDB), then only common users and the SYS user are returned.

If this view is queried from a pluggable database (PDB), only rows that pertain to the current PDB are returned.

| Column | Datatype | Description |
|---|---|---|
| USERID | NUMBER | User ID number |
| PASSW_EXPIRED | NUMBER | Indicates whether the password has expired (1) or not (0) |
| PASSW_IN_GRACE | NUMBER | Indicates whether the account is in grace (1) or not (0) |
| PASSW_LOCKED | NUMBER | Indicates whether the account is locked (1) or not (0) |
| PASSW_LOCK_UNLIM | NUMBER | Indicates whether the account is locked for an unlimited time (1) or not (0) |
| FAILED_LOGINS | NUMBER | The number of failed login attempts. The count is not cumulative; it is reset upon successful logon to the account |
| EXPIRATION_AFTER_GRACE | TIMESTAMP(3) | The expiration time after grace |
| PASSW_LOCK_TIME | TIMESTAMP(3) | The time the account was locked out |
| CON_ID | NUMBER | The ID of the container where the failed login occurred.<br>For users that are not common users, the CON_IDis the PDB ID where the failed login attempt occurred.<br>For common users, the CON_ID is 0.<br>The login attempts that occurred on a PDB are not displayed when you query V$RO_USER_ACCOUNT from another PDB. You only see the failed login attempts of any users (that are not |

| Column | Datatype | Description |
|---|---|---|
| | | common users) if those failed login attempts occurred on the same PDB from which you are querying V$RO_USER_ACCOUNT.<br>The failed login attempts of common users (and of the SYS user) are only displayed when V$RO_USER_ACCOUNT is queried from the root of a CDB, not when it is queried from a PDB.<br>In a non-CDB, the value is always 0. |

# V$ROLLNAME

V$ROLLNAME lists the names of all online rollback segments. It can only be accessed when the database is open.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| USN | NUMBER | | Rollback (undo) segment number |
| NAME | VARCHAR2(30) | NOT NULL | Rollback segment name |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ROLLSTAT

V$ROLLSTAT contains rollback segment statistics.

| Column | Datatype | Description |
|---|---|---|
| USN | NUMBER | Rollback segment number |
| LATCH | NUMBER | Latch for the rollback segment |
| EXTENTS | NUMBER | Number of extents in the rollback segment |
| RSSIZE | NUMBER | Size (in bytes) of the rollback segment. This value differs by the number of bytes in one database block from the value of the BYTES column of the *_SEGMENTS view.<br>**See Also:** *Oracle Database Administrator's Guide* for more information about space management for rollback segments |
| WRITES | NUMBER | Number of bytes written to the rollback segment |
| XACTS | NUMBER | Number of active transactions |
| GETS | NUMBER | Number of header gets |
| WAITS | NUMBER | Number of header waits |
| OPTSIZE | NUMBER | Optimal size of the rollback segment |

| Column | Datatype | Description |
|---|---|---|
| HWMSIZE | NUMBER | High watermark of the rollback segment size |
| SHRINKS | NUMBER | Number of times the size of a rollback segment decreases |
| WRAPS | NUMBER | Number of times rollback segment is wrapped |
| EXTENDS | NUMBER | Number of times rollback segment size is extended |
| AVESHRINK | NUMBER | Average shrink size |
| AVEACTIVE | NUMBER | Current size of active extents, averaged over time. |
| STATUS | VARCHAR2(15) | Rollback segment status:<br>• ONLINE<br>• PENDING OFFLINE<br>• OFFLINE<br>• FULL |
| CUREXT | NUMBER | Current extent |
| CURBLK | NUMBER | Current block |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ROWCACHE

V$ROWCACHE displays statistics for data dictionary activity. Each row contains statistics for one data dictionary cache.

| Column | Datatype | Description |
|---|---|---|
| CACHE# | NUMBER | Row cache ID number |
| TYPE | VARCHAR2(11) | Parent or subordinate row cache type |
| SUBORDINATE# | NUMBER | Subordinate set number |
| PARAMETER | VARCHAR2(32) | Name of the initialization parameter that determines the number of entries in the data dictionary cache |
| COUNT | NUMBER | Total number of entries in the cache |
| USAGE | NUMBER | Number of cache entries that contain valid data |
| FIXED | NUMBER | Number of fixed entries in the cache |
| GETS | NUMBER | Total number of requests for information on the data object |
| FASTGETS | NUMBER | Reserved for internal use |
| GETMISSES | NUMBER | Number of data requests resulting in cache misses |

| Column | Datatype | Description |
|---|---|---|
| SCANS | NUMBER | Number of scan requests |
| SCANMISSES | NUMBER | Number of times a scan failed to find the data in the cache |
| SCANCOMPLETES | NUMBER | For a list of subordinate entries, the number of times the list was scanned completely |
| MODIFICATIONS | NUMBER | Number of inserts, updates, and deletions |
| FLUSHES | NUMBER | Number of times flushed to disk |
| DLM_REQUESTS | NUMBER | Number of DLM requests |
| DLM_CONFLICTS | NUMBER | Number of DLM conflicts |
| DLM_RELEASES | NUMBER | Number of DLM releases |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ROWCACHE_PARENT

V$ROWCACHE_PARENT displays information for parent objects in the data dictionary. There is one row per lock owner, and one waiter for each object. This row shows the mode held or requested. For objects with no owners or waiters, a single row is displayed.

| Column | Datatype | Description |
|---|---|---|
| INDX | NUMBER | Index of the row |
| HASH | NUMBER | Hash value |
| ADDRESS | RAW(4 \| 8) | Address of the parent object |
| CACHE# | NUMBER | Parent cache ID |
| CACHE_NAME | VARCHAR2(64) | Parent cache name |
| EXISTENT | VARCHAR2(1) | Indicates whether the object is an existing object |
| LOCK_MODE | NUMBER | Mode the lock is held in |
| LOCK_REQUEST | NUMBER | Mode the lock is requested in |
| TXN | RAW(4 \| 8) | Transaction currently locking the object |
| SADDR | RAW(4 \| 8) | Address of the session |
| INST_LOCK_REQUEST | NUMBER | Mode in which instance lock is being requested. This column is only relevant for Real Application Clusters. |

| Column | Datatype | Description |
|---|---|---|
| INST_LOCK_RELEASE | NUMBER | Whether the instance lock needs to be released. This column is only relevant for Real Application Clusters. |
| INST_LOCK_TYPE | VARCHAR2(2) | Type of instance lock. This column is only relevant for Real Application Clusters. |
| INST_LOCK_ID1 | RAW(4) | ID associated with the instance lock. This column is only relevant for Real Application Clusters. |
| INST_LOCK_ID2 | RAW(4) | ID associated with the instance lock. This column is only relevant for Real Application Clusters. |
| KEY | RAW(100) | Contents of the key. This column is only relevant for Real Application Clusters. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$ROWCACHE_SUBORDINATE

V$ROWCACHE_SUBORDINATE displays information for subordinate objects in the data dictionary.

| Column | Datatype | Description |
|---|---|---|
| INDX | NUMBER | The index |
| HASH | NUMBER | The hash value |
| ADDRESS | RAW(4 \| 8) | Address of the subordinate object |
| CACHE# | NUMBER | The parent cache ID |
| SUBCACHE# | NUMBER | The subcache ID |
| SUBCACHE_NAME | VARCHAR2(64) | The subcache name |
| EXISTENT | VARCHAR2(1) | Whether the object is an existing object |
| PARENT | RAW(4 \| 8) | Address of the parent object |
| KEY | RAW(100) | The contents of the key |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RSRC_CONS_GROUP_HISTORY

V$RSRC_CONS_GROUP_HISTORY displays a history of consumer group statistics for each entry in V$RSRC_PLAN_HISTORY that has a non-NULL plan.

When the STATISTICS_LEVEL is set to TYPICAL or ALL, this view contains information about CPU utilization and wait times even when no Resource Manager plan is set or when the Resource Manager plan does not monitor CPU or session resources.

| Column | Datatype | Description |
|---|---|---|
| SEQUENCE# | NUMBER | A sequential counter that uniquely describes the V $RSRC_PLAN_HISTORY entry to which these consumer group statistics apply. When the instance is restarted, this value is reset to zero. |
| ID | NUMBER | Consumer group object ID (a unique number, consistent across database shutdowns and startups) |
| NAME | VARCHAR2(30) | Name of the consumer group |
| REQUESTS | NUMBER | Cumulative number of requests that were executed in the consumer group |
| CPU_WAIT_TIME | NUMBER | Cumulative amount of time that sessions waited for CPU because of resource management. This does not include waits due to latch or enqueue contention, I/O waits, and so on. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_WAITS | NUMBER | Cumulative number of times all sessions in the consumer group had to wait for CPU because of resource management. This does not include waits due to latch or enqueue contention, I/O waits, and so on. When CPU resources are not being actively managed, this value is set to zero. |
| CONSUMED_CPU_TIME | NUMBER | Cumulative amount of CPU time consumed by all sessions in the consumer group (in milliseconds) |
| YIELDS | NUMBER | Cumulative number of times that sessions in the consumer group had to yield CPU to other sessions because of quantum expiration. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_DECISIONS | NUMBER | Percentage of CPU decisions for which the consumer group was present. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_DECISIONS_EXCLUSIVE | NUMBER | Percentage of the CPU decisions for which the consumer group was present and was the only consumer group present. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_DECISIONS_WON | NUMBER | Percentage of the CPU decisions that the consumer group won. When CPU resources are not being actively managed, this value is set to zero. |

| Column | Datatype | Description |
|---|---|---|
| ACTIVE_SESS_LIMIT_HIT | NUMBER | Number of times that sessions in the consumer group were queued because the consumer group reached its active session limit |
| UNDO_LIMIT_HIT | NUMBER | Number of times that queries in the consumer group were canceled because the consumer group reached its UNDO_POOL limit |
| SWITCHES_IN_CPU_TIME | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_TIME limit |
| SWITCHES_OUT_CPU_TIME | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_TIME limit |
| SWITCHES_IN_IO_MEGABYTES | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_MEGABYTES limit |
| SWITCHES_OUT_IO_MEGABYTES | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_MEGABYTES limit |
| SWITCHES_IN_IO_REQUESTS | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_REQS limit |
| SWITCHES_OUT_IO_REQUESTS | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_REQS limit |
| SWITCHES_IN_IO_LOGICAL | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_LOGICAL limit |
| SWITCHES_OUT_IO_LOGICAL | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_LOGICAL limit |
| SWITCHES_IN_ELAPSED_TIME | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_ELAPSED_TIME limit |
| SWITCHES_OUT_ELAPSED_TIME | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_ELAPSED_TIME limit |
| SQL_CANCELED | NUMBER | Number of times that SQL queries running in the consumer group were aborted because they exceeded the Resource Manager plan's SWITCH_TIME limit and CANCEL_SQL was specified as the Resource Manager plan's SWITCH_GROUP |
| ACTIVE_SESS_KILLED | NUMBER | Number of times that sessions running in the consumer group were terminated because they exceeded the Resource Manager plan's SWITCH_TIME limit and KILL_SESSION was specified as the Resource Manager plan's SWITCH_GROUP |
| IDLE_SESS_KILLED | NUMBER | Number of times that sessions in the consumer group were killed because they were idle for too long (reached MAX_IDLE_TIME) |
| IDLE_BLKR_SESS_KILLED | NUMBER | Number of times that sessions in the consumer group were killed because they were idle too long (reached MAX_IDLE_BLOCKER_TIME) and were blocking other sessions |
| QUEUED_TIME | NUMBER | Total amount of time that sessions in the consumer group have spent in the QUEUED state because of the active session limit (in milliseconds) |

| Column | Datatype | Description |
|---|---|---|
| QUEUE_TIME_OUTS | NUMBER | Number of times that requests from sessions in the consumer group timed out because they were queued for too long (reached QUEUEING_P1) |
| IO_SERVICE_TIME | NUMBER | Cumulative I/O wait time (in milliseconds) |
| IO_SERVICE_WAITS | NUMBER | Total number of wait requests |
| SMALL_READ_MEGABYTES | NUMBER | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | Number of multiblock megabytes written |
| SMALL_READ_REQUESTS | NUMBER | Number of single block read requests |
| SMALL_WRITE_REQUESTS | NUMBER | Number of single block write requests |
| LARGE_READ_REQUESTS | NUMBER | Number of multiblock read requests |
| LARGE_WRITE_REQUESTS | NUMBER | Number of multiblock write requests |
| PQS_COMPLETED | NUMBER | Total number of completed parallel statements in the consumer group |
| PQ_SERVERS_USED | NUMBER | Total number of parallel servers used by completed parallel statements in the consumer group |
| PQS_QUEUED | NUMBER | Number of times that sessions in the consumer group were queued when trying to run parallel statements |
| PQ_ACTIVE_TIME | NUMBER | Cumulative sum of the parallel active times for all completed parallel statements in the consumer group (in milliseconds) |
| PQ_QUEUED_TIME | NUMBER | Total amount of time that sessions in the consumer group were queued when trying to run parallel statements (in milliseconds) |
| PQ_QUEUE_TIME_OUTS | NUMBER | Number of times that parallel statements from sessions in the consumer group timed out because their queue time exceeded the Resource Manager plan's PARALLEL_QUEUE_TIMEOUT limit |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$RSRC_PLAN_HISTORY"
- "STATISTICS_LEVEL"

# V$RSRC_CONSUMER_GROUP

V$RSRC_CONSUMER_GROUP displays data related to currently active resource consumer groups.

When the STATISTICS_LEVEL is set to TYPICAL or ALL, this view contains information about CPU utilization and wait times even when no Resource Manager plan is set or when the Resource Manager plan does not monitor CPU or session resources.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Consumer group object ID (a unique number, consistent across database shutdowns and startups) |
| NAME | VARCHAR2(32) | Name of the consumer group |
| ACTIVE_SESSIONS | NUMBER | Number of currently active sessions in the consumer group |
| EXECUTION_WAITERS | NUMBER | Number of currently active sessions waiting for an execution time slice in which they will be able to use CPU |
| REQUESTS | NUMBER | Cumulative number of requests that were executed in the consumer group |
| CPU_WAIT_TIME | NUMBER | Cumulative amount of time that sessions waited for CPU because of resource management. This does not include waits due to latch or enqueue contention, I/O waits, and so on. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_WAITS | NUMBER | Cumulative number of times all sessions in the consumer group had to wait for CPU because of resource management. This does not include waits due to latch or enqueue contention, I/O waits, and so on. When CPU resources are not being actively managed, this value is set to zero. |
| CONSUMED_CPU_TIME | NUMBER | Cumulative amount of CPU time consumed by all sessions in the consumer group (in milliseconds) |
| YIELDS | NUMBER | Cumulative number of times that sessions in the consumer group had to yield CPU to other sessions because of quantum expiration. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_DECISIONS | NUMBER | Percentage of CPU decisions for which the consumer group was present. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_DECISIONS_EXCLUSIVE | NUMBER | Percentage of the CPU decisions for which the consumer group was present and was the only consumer group present. When CPU resources are not being actively managed, this value is set to zero. |

| Column | Datatype | Description |
|---|---|---|
| CPU_DECISIONS_WON | NUMBER | Percentage of the CPU decisions that the consumer group won. When CPU resources are not being actively managed, this value is set to zero. |
| QUEUE_LENGTH | NUMBER | Number of sessions waiting in the queue |
| CURRENT_UNDO_CONSUMPTION | NUMBER | Current amount (in KB) of undo consumed by the consumer group |
| ACTIVE_SESSION_LIMIT_HIT | NUMBER | Number of times that sessions in the consumer group were queued because the consumer group reached its active session limit |
| UNDO_LIMIT_HIT | NUMBER | Number of times that queries in the consumer group were canceled because the consumer group reached its UNDO_POOL limit |
| SWITCHES_IN_CPU_TIME | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_TIME limit |
| SWITCHES_OUT_CPU_TIME | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_TIME limit |
| SWITCHES_IN_IO_MEGABYTES | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_MEGABYTES limit |
| SWITCHES_OUT_IO_MEGABYTES | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_MEGABYTES limit |
| SWITCHES_IN_IO_REQUESTS | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_REQS limit |
| SWITCHES_OUT_IO_REQUESTS | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_REQS limit |
| SWITCHES_IN_IO_LOGICAL | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_IO_LOGICAL limit |
| SWITCHES_OUT_IO_LOGICAL | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_IO_LOGICAL limit |
| SWITCHES_IN_ELAPSED_TIME | NUMBER | Number of switches into the consumer group because of the Resource Manager plan's SWITCH_ELAPSED_TIME limit |
| SWITCHES_OUT_ELAPSED_TIME | NUMBER | Number of switches out of the consumer group because of the Resource Manager plan's SWITCH_ELAPSED_TIME limit |
| SQL_CANCELED | NUMBER | Number of times that SQL queries running in the consumer group were aborted because they exceeded the Resource Manager plan's SWITCH_TIME limit and CANCEL_SQL was specified as the Resource Manager plan's SWITCH_GROUP |
| ACTIVE_SESSIONS_KILLED | NUMBER | Number of times that sessions running in the consumer group were terminated because they exceeded the Resource Manager plan's SWITCH_TIME limit and KILL_SESSION was specified as the Resource Manager plan's SWITCH_GROUP |
| IDLE_SESSIONS_KILLED | NUMBER | Number of times that sessions in the consumer group were killed because they were idle for too long (reached MAX_IDLE_TIME) |

| Column | Datatype | Description |
|---|---|---|
| IDLE_BLKR_SESSIONS_KILLED | NUMBER | Number of times that sessions in the consumer group were killed because they were idle too long (reached MAX_IDLE_BLOCKER_TIME) and were blocking other sessions |
| QUEUED_TIME | NUMBER | Total amount of time that sessions in the consumer group have spent in the QUEUED state because of the active session limit (in milliseconds) |
| QUEUE_TIME_OUTS | NUMBER | Number of times that requests from sessions in the consumer group timed out because they were queued for too long (reached QUEUEING_P1) |
| IO_SERVICE_TIME | NUMBER | Cumulative I/O wait time (in milliseconds) |
| IO_SERVICE_WAITS | NUMBER | Total number of wait requests |
| SMALL_READ_MEGABYTES | NUMBER | Number of single block megabytes read |
| SMALL_WRITE_MEGABYTES | NUMBER | Number of single block megabytes written |
| LARGE_READ_MEGABYTES | NUMBER | Number of multiblock megabytes read |
| LARGE_WRITE_MEGABYTES | NUMBER | Number of multiblock megabytes written |
| SMALL_READ_REQUESTS | NUMBER | Number of single block read requests |
| SMALL_WRITE_REQUESTS | NUMBER | Number of single block write requests |
| LARGE_READ_REQUESTS | NUMBER | Number of multiblock read requests |
| LARGE_WRITE_REQUESTS | NUMBER | Number of multiblock write requests |
| CURRENT_PQS_ACTIVE | NUMBER | Number of active parallel statements in the consumer group. This value does not include parallel statements that are never queued, such as GV$ queries. |
| CURRENT_PQ_SERVERS_ACTIVE | NUMBER | Number of active parallel servers in the consumer group. This value does not include servers running parallel statements that are never queued, such as GV$ queries. |
| PQS_QUEUED | NUMBER | Number of times that sessions in the consumer group were queued when trying to run parallel statements |
| PQS_COMPLETED | NUMBER | Total number of completed parallel statements in the consumer group |
| PQ_SERVERS_USED | NUMBER | Total number of parallel servers used by completed parallel statements in the consumer group |
| PQ_ACTIVE_TIME | NUMBER | Cumulative sum of the parallel active times for all completed parallel statements in the consumer group (in milliseconds) |

| Column | Datatype | Description |
|---|---|---|
| CURRENT_PQS_QUEUED | NUMBER | Number of sessions in the consumer group that are waiting in the parallel statement queue trying to run parallel statements |
| PQ_QUEUED_TIME | NUMBER | Total amount of time that sessions in the consumer group were queued when trying to run parallel statements (in milliseconds) |
| PQ_QUEUE_TIME_OUTS | NUMBER | Number of times that parallel statements from sessions in the consumer group timed out because their queue time exceeded the Resource Manager plan's PARALLEL_QUEUE_TIMEOUT limit |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "STATISTICS_LEVEL"
- *Oracle Database Administrator's Guide* for information on resource groups
- *Oracle Database PL/SQL Packages and Types Reference* for information on creating resource groups with the DBMS_RESOURCE_MANAGER package

# V$RSRC_CONSUMER_GROUP_CPU_MTH

V$RSRC_CONSUMER_GROUP_CPU_MTH displays all resource allocation methods defined for resource consumer groups.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(40) | Name of the CPU resource allocation method |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- *Oracle Database Administrator's Guide* for information on resource allocation methods
- *Oracle Database PL/SQL Packages and Types Reference* on defining resource allocation methods for consumer groups with the DBMS_RESOURCE_MANAGER package
- "V$RSRC_PLAN_CPU_MTH" for a listing of all resource allocation methods defined for resource plans

# V$RSRC_PLAN

V$RSRC_PLAN displays the names of all currently active resource plans.

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Resource plan ID (a unique number, consistent across database shutdowns and startups). This is also the data dictionary object ID. |
| NAME | VARCHAR2(32) | Name of the resource plan |
| IS_TOP_PLAN | VARCHAR2(5) | Indicates whether the resource plan is the current top plan (TRUE) or whether the resource plan is a subplan of the current top plan (FALSE) |
| CPU_MANAGED | VARCHAR2(3) | Indicates whether the resource plan has parameters that specify a policy for how the Resource Manager should schedule sessions to manage CPU usage (ON) or whether Resource Manager is not managing CPU usage (OFF) |
| INSTANCE_CAGING | VARCHAR2(3) | Indicates whether instance caging is enabled (ON) or disabled (OFF). Instance caging is enabled if the CPU_COUNT initialization parameter is set to a value other than 0 and Resource Manager is enabled. |
| PARALLEL_SERVERS_ACTIVE | NUMBER | Total number of active parallel servers on the instance |
| PARALLEL_SERVERS_TOTAL | NUMBER | The value of PARALLEL_SERVERS_TARGET for the instance. Parallel statements are queued if the total number of active parallel servers exceeds this value. |
| PARALLEL_EXECUTION_MANAGED | VARCHAR2(32) | State of parallel statement queuing:<br>• OFF - Parallel statement queuing is disabled<br>• STARTUP - Parallel statement queuing is enabled. This is a temporary state that can occur when an Oracle RAC database is undergoing configuration changes<br>• FIFO - Parallel statement queuing is enabled. All parallel statements are managed in a single Oracle RAC FIFO queue<br>• FULL - Parallel statement queuing is enabled. All parallel statements are managed in per-consumer group queues according to the current resource plan. This state is used when a resource plan that contains resource allocation directives (MGMT_P*) is enabled. |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| DIRECTIVE_TYPE | VARCHAR2(32) | For internal use only |
| SHARES | NUMBER | For internal use only |
| UTILIZATION_LIMIT | NUMBER | For internal use only |
| PARALLEL_SERVER_LIMIT | NUMBER | For internal use only |
| MEMORY_MIN | NUMBER | For internal use only |
| MEMORY_LIMIT | NUMBER | For internal use only |
| PROFILE | VARCHAR2(32) | For internal use only |

The resource plan with CON_ID=ROOT is the CDB resource plan.

**See Also:**

- "DBA_RSRC_PLANS" for a listing of all plans in the database
- "DBA_CDB_RSRC_PLANS" for information about CDB resource plans
- "DBA_CDB_RSRC_PLAN_DIRECTIVES" for information about CDB resource plan directives
- *Oracle Database Administrator's Guide* for information on resource plans
- *Oracle Database PL/SQL Packages and Types Reference* for information on defining resource allocation methods for consumer groups with the DBMS_RESOURCE_MANAGER package

# V$RSRC_PLAN_CPU_MTH

V$RSRC_PLAN_CPU_MTH displays all available CPU resource allocation methods defined for resource plans.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(40) | Name of the resource allocation method |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$RSRC_CONSUMER_GROUP_CPU_MTH" for a listing of resource allocation methods defined for consumer groups
- *Oracle Database Administrator's Guide* for information on resource plans
- *Oracle Database PL/SQL Packages and Types Reference* for information on defining resource allocation methods for consumer plans with the DBMS_RESOURCE_MANAGER package

# V$RSRC_PLAN_HISTORY

V$RSRC_PLAN_HISTORY displays a history of when a resource plan was enabled, disabled, or modified on the instance. Up to 15 of the most recent entries are shown. Once the database is opened, this view shows at least one row. The row with the most recent START_TIME and with END_TIME equal to NULL gives information about the current resource plan.

| Column | Datatype | Description |
|---|---|---|
| SEQUENCE# | NUMBER | A sequential counter that uniquely describes a row. When the instance is restarted, this value is reset to zero. |
| ID | NUMBER | Resource plan ID; NULL if the Resource Manager was disabled |
| NAME | VARCHAR2(30) | Resource plan name; NULL if the Resource Manager was disabled |
| START_TIME | DATE | Time that the resource plan was enabled |
| END_TIME | DATE | Time that the resource plan was disabled; NULL if the row contains the current resource plan information |
| ENABLED_BY_SCHEDULER | VARCHAR2(5) | Indicates whether the plan was enabled by a Job Scheduler window (TRUE) or not (FALSE) |
| WINDOW_NAME | VARCHAR2(128) | Job Scheduler window that triggered the resource plan event; NULL if a user triggered the resource plan event |
| ALLOWED_AUTOMATED_SWITCHES | VARCHAR2(5) | Indicates whether automated plan switches were allowed after this resource plan event (TRUE) or whether automated plan switches were disabled after this resource plan event (FALSE)<br>**See Also:** the SWITCH_PLAN procedure |
| CPU_MANAGED | VARCHAR2(3) | Indicates whether the resource plan has parameters that specify a policy for how the Resource Manager should schedule sessions to manage CPU usage (ON) or whether Resource Manager is not managing CPU usage (OFF) |

| Column | Datatype | Description |
|---|---|---|
| INSTANCE_CAGING | VARCHAR2(3) | Indicates whether instance caging is enabled (ON) or disabled (OFF). Instance caging is enabled if the CPU_COUNT initialization parameter is set to a value other than 0 and Resource Manager is enabled. |
| PARALLEL_EXECUTION_MANAGED | VARCHAR2(32) | State of parallel statement queuing:<br>• OFF - Parallel statement queuing is disabled<br>• STARTUP - Parallel statement queuing is enabled. This is a temporary state that can occur when an Oracle RAC database is undergoing configuration changes<br>• FIFO - Parallel statement queuing is enabled. All parallel statements are managed in a single Oracle RAC FIFO queue<br>• FULL - Parallel statement queuing is enabled. All parallel statements are managed in per-consumer group queues according to the current resource plan. This state is used when a resource plan that contains resource allocation directives (MGMT_P*) is enabled. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RSRC_SESSION_INFO

V$RSRC_SESSION_INFO displays Resource Manager statistics per session.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |
| CURRENT_CONSUMER_GROUP_ID | NUMBER | Object ID of the consumer group in which the session currently belongs; NULL if the session has not yet logged in |
| CURRENT_CONSUMER_GROUP | VARCHAR2(32) | The name of the consumer group in which the session currently belongs |
| ORIG_CONSUMER_GROUP_ID | NUMBER | Object ID of the consumer group in which the session was placed by the consumer group mappings; NULL if the session has not yet logged in<br>This group may not be the current group because the SWITCH_GROUP directive in the current plan may have changed the session's current group. This group may not equal the MAPPED_CONSUMER_GROUP because the MAPPED_CONSUMER_GROUP may not be part of the current plan. |
| MAPPING_ATTRIBUTE | VARCHAR2(32) | Session attribute that was used to map the session into its original consumer group; NULL if no mapping was used<br>**See Also:** "DBA_RSRC_GROUP_MAPPINGS" for more details |
| MAPPED_CONSUMER_GROUP | VARCHAR2(32) | Consumer group to which the session was originally mapped; NULL if no mapping was used |

| Column | Datatype | Description |
|---|---|---|
| | | This may not correspond to the original consumer group, because the mapped group may not be part of the current plan.<br>**See Also:** "DBA_RSRC_GROUP_MAPPINGS" for more details |
| STATE | VARCHAR2(32) | Current state of the session:<br>• NOT MANAGED - Session has not logged in or the current Resource Manager plan does not require the session to be managed at this point in time<br>• RUNNING - Session is currently running on the CPU<br>• WAITING FOR CPU - Session is ready to run. It is waiting for a CPU quantum to run.<br>• QUEUED - Session is queued because the active session limit was reached<br>• IDLE - Session is idle<br>• IDLE BLKR - Session is idle and blocking another session<br>• WAITING - Session is currently in a wait<br>**See Also:** "V$SESSION_WAIT" for the wait type<br>• WAITING_FOR_IO - Session is waiting to submit an I/O request<br>• UNBOUND - Session is not bound to any process<br>• EXITING - Session is about to terminate |
| ACTIVE | VARCHAR2(5) | Indicates whether the session is currently active (TRUE) or not (FALSE). This includes when one of the following conditions is true:<br>• Session is in the top call<br>• Session has a transaction in progress<br>• Session is using temporary space objects<br>• Session holds user enqueues |
| CURRENT_IDLE_TIME | NUMBER | Number of seconds the session has been idle (in states IDLE or IDLE BLKR) while in this consumer group; NULL if the current Resource Manager plan does not require updating this statistic. This value is reset to zero when the session becomes active. |
| CURRENT_CPU_WAIT_TIME | NUMBER | Amount of time (in milliseconds) the session has waited for CPU because of resource management (in state WAIT FOR CPU) while in the current consumer group; NULL if the current Resource Manager plan does not require updating this statistic. This does not include waits due to latch or enqueue contention, I/O waits, and so on. If SWITCH_TIME_IN_CALL is used, then this will be reset at the end of every call. |
| CPU_WAIT_TIME | NUMBER | Cumulative amount of time (in milliseconds) the session has waited for CPU (in its lifetime) because of resource management. This does not include waits due to latch or enqueue contention, I/O waits, and so on. |
| CURRENT_CPU_WAITS | NUMBER | Number of times the session had to wait for CPU because of resource management while in this consumer group; NULL if the current Resource Manager plan does not require updating this statistic. This does not include waits due to latch or enqueue contention, I/O waits, and so on. If SWITCH_TIME_IN_CALL is used, then this will be reset at the end of every call. |
| CPU_WAITS | NUMBER | Cumulative number of times the session had to wait for CPU (in its lifetime) because of resource management. This does not |

| Column | Datatype | Description |
|---|---|---|
| | | include waits due to latch or enqueue contention, I/O waits, and so on. |
| CURRENT_CONSUMED_CPU_TIME | NUMBER | Amount of CPU time (in milliseconds) consumed by the session while in the current consumer group; NULL if the current Resource Manager plan does not require updating this statistic. If SWITCH_TIME_IN_CALL is used, then this will be reset at the end of every call. |
| CONSUMED_CPU_TIME | NUMBER | Cumulative amount of CPU time consumed by the session (in its lifetime) (in milliseconds) |
| CURRENT_ACTIVE_TIME | NUMBER | Amount of time (in milliseconds) the session has been active while in the current consumer group; NULL if the current Resource Manager plan does not require updating this statistic. Active time is time spent running and waiting while executing a call. It does not include the time a session waited for CPU resources. If SWITCH_TIME_IN_CALL is used, then this is reset at the end of every call. |
| ACTIVE_TIME | NUMBER | Cumulative amount of active time (in milliseconds) consumed by the session (in its lifetime) |
| CURRENT_QUEUED_TIME | NUMBER | Amount of time (in milliseconds) the current request from the session has been queued (in state QUEUED). If the session does not have a request currently queued up, then this number will be zero. |
| QUEUED_TIME | NUMBER | Total amount of time (in milliseconds) the session has spent in the QUEUED state (in its lifetime) |
| CURRENT_YIELDS | NUMBER | Number of times the session had to yield the CPU to other sessions (due to quantum expiration) while in the current consumer group; NULL if the current Resource Manager plan does not require updating this statistic. If SWITCH_TIME_IN_CALL is used, then this is reset at the end of every call. |
| YIELDS | NUMBER | Cumulative number of times the session had to yield CPU to other sessions due to quantum expiration (in its lifetime) |
| CURRENT_UNDO_CONSUMPTION | NUMBER | Current amount (in KB) of undo consumed by the session; NULL if the current Resource Manager plan does not have an UNDO_POOL directive |
| MAX_UNDO_CONSUMPTION | NUMBER | Maximum amount of undo consumed (in KB) during the session's lifetime. This value may not be updated because the current Resource Manager plan may not have an UNDO_POOL directive. |
| SQL_CANCELED | NUMBER | Number of times SQL queries running in the session were canceled due to exceeding the Resource Manager plan's SWITCH_TIME limit. |
| QUEUE_TIME_OUTS | NUMBER | Number of times requests from the session timed out because they queued longer than the Resource Manager plan's limit |
| ESTIMATED_EXECUTION_LIMIT_HIT | NUMBER | Number of times requests from the session were not run because the optimizer's estimated time to execute the query exceeded the MAX_EST_EXEC_TIME limit |

| Column | Datatype | Description |
|---|---|---|
| CURRENT_IO_SERVICE_TIME | NUMBER | Current I/O wait time of the session (in milliseconds) for the current SQL operation |
| IO_SERVICE_TIME | NUMBER | Cumulative amount of I/O wait time by the session (in its lifetime) (in milliseconds) |
| CURRENT_IO_SERVICE_WAITS | NUMBER | Current I/O waits by session for the current SQL operation |
| IO_SERVICE_WAITS | NUMBER | Cumulative I/O waits by session (in its lifetime) |
| CURRENT_SMALL_READ_MEGABYTES | NUMBER | Number of single block megabytes read by the session for the current SQL operation |
| SMALL_READ_MEGABYTES | NUMBER | Total number of single block megabytes read by the session (in its lifetime) |
| CURRENT_LARGE_READ_MEGABYTES | NUMBER | Number of multiblock megabytes read by the session for the current SQL operation |
| LARGE_READ_MEGABYTES | NUMBER | Total number of multiblock megabytes read by the session (in its lifetime) |
| CURRENT_SMALL_WRITE_MEGABYTES | NUMBER | Number of single block megabytes written by the session for the current SQL operation |
| SMALL_WRITE_MEGABYTES | NUMBER | Total number of single block megabytes written by the session (in its lifetime) |
| CURRENT_LARGE_WRITE_MEGABYTES | NUMBER | Number of multiblock megabytes written by the session for the current SQL operation |
| LARGE_WRITE_MEGABYTES | NUMBER | Total number of multiblock megabytes written by the session (in its lifetime) |
| CURRENT_SMALL_READ_REQUESTS | NUMBER | Number of single block read requests by the session for the current SQL operation |
| SMALL_READ_REQUESTS | NUMBER | Total number of single block read requests by the session (in its lifetime) |
| CURRENT_SMALL_WRITE_REQUESTS | NUMBER | Number of single block write requests by the session for the current SQL operation |
| SMALL_WRITE_REQUESTS | NUMBER | Total number of single block write requests by the session (in its lifetime) |
| CURRENT_LARGE_READ_REQUESTS | NUMBER | Number of multiblock read requests by the session for the current SQL operation |
| LARGE_READ_REQUESTS | NUMBER | Total number of multiblock read requests by the session (in its lifetime) |
| CURRENT_LARGE_WRITE_REQUESTS | NUMBER | Number of multiblock write requests by the session for the current SQL operation |
| LARGE_WRITE_REQUESTS | NUMBER | Total number of multiblock write requests by the session (in its lifetime) |

| Column | Datatype | Description |
|---|---|---|
| CURRENT_PQ_ACTIVE_TIME | NUMBER | Amount of time that the current active parallel statement has been executing for the current SQL operation, not including the amount of time that the statement has been queued (in milliseconds). If the parallel statement is queued, then the value is 0. |
| PQ_ACTIVE_TIME | NUMBER | Cumulative amount of time that parallel statements have been executed over the lifetime of the session (in milliseconds) |
| DOP | NUMBER | Degree of parallelism for the active or queued parallel statement, if there are any in the session |
| PQ_SERVERS | NUMBER | The number of active parallel servers if the session is active and running the parallel query. If the query is queued, the number of parallel servers that this query is trying to run with is shown. |
| ESTIMATED_EXECUTION_TIME | NUMBER | Estimated execution time for the parallel statement, as estimated by the optimizer (in milliseconds). You can compare this value to CURRENT_PQ_ACTIVE_TIME to estimate how much longer the parallel statement will run. |
| CURRENT_PQ_QUEUED_TIME | NUMBER | Amount of time that the current parallel statement in the session has been queued (in milliseconds) for the current SQL operation. If the session does not have a queued parallel statement, then the value is 0. |
| PQ_QUEUED_TIME | NUMBER | Total amount of time that the session has spent in the PQ_QUEUED state in its lifetime (in milliseconds) |
| PQ_QUEUED | NUMBER | Number of times that parallel statements in the session have been queued |
| PQ_QUEUE_TIME_OUTS | NUMBER | Number of times that parallel statements in the session timed out because their queue time exceeded the Resource Manager plan's PARALLEL_QUEUE_TIMEOUT limit |
| PQ_ACTIVE | VARCHAR2(5) | Indicates whether the session is actively running a parallel statement (TRUE) or not (FALSE). |
| PQ_STATUS | VARCHAR2(44) | The status of the parallel statement running in this session:<br>• Active: The parallel statement is currently active and running<br>• Queued: The parallel statement is queued and is not a possible candidate to be run next<br>• Queue head: The parallel statement is queued due to the global systemwide limit and is the next parallel statement to be dequeued and run<br>• Queue head - waiting on CG limit: The parallel statement is queued due to the consumer group limit and is the one to be run next from the session's consumer group<br>• Queue head - waiting on service availability: The parallel statement is queued since there are not enough parallel servers available for the service this session is running on. This session is the one to be run next from the session's service<br>• Queue head - waiting on PDB limit: The parallel statement is queued due to the PDB limit and is the one to be run next from the session's PDB |

| Column | Datatype | Description |
|---|---|---|
| | | • NULL: The session currently does not involve a parallel execution |
| CURRENT_LOGICAL_IOS | NUMBER | Number of logical I/O requests by the session for the current SQL operation |
| LOGICAL_IOS | NUMBER | Total number of logical I/O requests in this session's lifetime |
| CURRENT_ELAPSED_TIME | NUMBER | Elapsed time of the session's current SQL operation |
| ELAPSED_TIME | NUMBER | Total elapsed time for all of this session's SQL operations in its lifetime |
| LAST_ACTION | VARCHAR2(48) | The most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following:<br>• CANCEL_SQL<br>• KILL_SESSION<br>• LOG_ONLY<br>• SWITCH TO <CG NAME><br>For the last value, <CG NAME> is the name of the consumer group that the SQL operation was switched to. If the Resource Plan has since been changed then <CG NAME> is the ID of the consumer group. |
| LAST_ACTION_REASON | VARCHAR2(30) | The reason for the most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following:<br>• SWITCH_CPU_TIME<br>• SWITCH_IO_REQS<br>• SWITCH_IO_MBS<br>• SWITCH_ELAPSED_TIME<br>• SWITCH_IO_LOGICAL |
| LAST_ACTION_TIME | DATE | The time of the most recent action that was taken on this SQL operation by Resource Manager |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RSRCMGRMETRIC

V$RSRCMGRMETRIC displays information about resources consumed and wait times per consumer group.

When the STATISTICS_LEVEL is set to TYPICAL or ALL, this view contains information about CPU utilization and wait times even when no Resource Manager plan is set or when the Resource Manager plan does not monitor CPU or session resources. Metrics are collected and stored every minute when CPU utilization is not being monitored.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time for the metric value |
| END_TIME | DATE | End time for the metric value |
| INTSIZE_CSEC | NUMBER | Size of the time period |
| SEQUENCE# | NUMBER | A sequential counter that uniquely describes the V$RSRC_PLAN_HISTORY entry to which these consumer group statistics apply. When the instance is restarted, this value is reset to zero. |
| CONSUMER_GROUP_ID | NUMBER | Consumer group object ID (a unique number, consistent across database shutdowns and startups) |
| CONSUMER_GROUP_NAME | VARCHAR2(30) | Name of the consumer group |
| CPU_CONSUMED_TIME | NUMBER | Cumulative amount of CPU time consumed by all sessions in the consumer group, in milliseconds |
| CPU_WAIT_TIME | NUMBER | Cumulative amount of time that sessions waited for CPU because of resource management, in milliseconds. This does not include waits due to latch or enqueue contention, I/O waits, and so on. When CPU resources are not being actively managed, this value is set to zero. |
| NUM_CPUS | NUMBER | Number of CPUs that the Resource Manager is utilizing. If instance caging is enabled, then this column is equal to the value of the CPU_COUNT initialization parameter. If instance caging is not enabled, then this column is equal to the total number of CPUs in the system. |
| RUNNING_SESSIONS_LIMIT | NUMBER | Maximum number of sessions in the consumer group that can run simultaneously. The value of this column is set by the Resource Manager plan's MAX_UTILIZATION_LIMIT directive. |
| AVG_RUNNING_SESSIONS | NUMBER | Average number of sessions in the consumer group that are currently running |
| AVG_WAITING_SESSIONS | NUMBER | Average number of sessions in the consumer group that are waiting for CPU due to resource management. When CPU resources are not being actively managed, this value is set to zero. |
| CPU_UTILIZATION_LIMIT | NUMBER | Maximum percentage of CPU that the consumer group can use at any time. The value of this column is set by the Resource Manager plan's MAX_UTILIZATION_LIMIT directive. If instance caging is enabled, then this value is derived using the CPU_COUNT initialization parameter. |
| AVG_CPU_UTILIZATION | NUMBER | Average percentage CPU consumed by the consumer group |
| CPU_DECISIONS | NUMBER | Percentage of CPU decisions for which the consumer group was present. When CPU resources are not being actively managed, this value is set to zero. This column is deprecated. |
| CPU_DECISIONS_EXCLUSIVE | NUMBER | Percentage of the CPU decisions for which the consumer group was present and was the only consumer group present. When |

| Column | Datatype | Description |
|---|---|---|
| | | CPU resources are not being actively managed, this value is set to zero. This column is deprecated. |
| CPU_DECISIONS_WON | NUMBER | Percentage of the CPU decisions that the consumer group won. When CPU resources are not being actively managed, this value is set to zero. This column is deprecated. |
| IO_REQUESTS | NUMBER | I/O requests |
| IO_MEGABYTES | NUMBER | I/O megabytes |
| AVG_ACTIVE_PARALLEL_STMTS | NUMBER | The average number of parallel statements that were running during the 1-minute metric window |
| AVG_QUEUED_PARALLEL_STMTS | NUMBER | The average number of parallel statements that were queued during the 1-minute metric window |
| AVG_ACTIVE_PARALLEL_SERVERS | NUMBER | The average number of parallel servers that were actively running as part of a parallel statement during the 1-minute metric window |
| AVG_QUEUED_PARALLEL_SERVERS | NUMBER | The average number of parallel servers that were requested by queued parallel statements during the 1-minute metric window |
| PARALLEL_SERVERS_LIMIT | NUMBER | The number of parallel servers allowed to be used by this consumer group during the 1-minute metric window |
| PLAN_NAME | VARCHAR2(30) | Resource Manager plan name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"STATISTICS_LEVEL"

# V$RSRCMGRMETRIC_HISTORY

V$RSRCMGRMETRIC_HISTORY displays a history (the last one hour) of resource manager metrics, taken from V$RSRCMGRMETRIC. When a resource plan is set, this history is cleared and restarted. This view provides information about resources consumed and wait times per consumer group.

The columns for V$RSRCMGRMETRIC_HISTORY are the same as those for V$RSRCMGRMETRIC.

**See Also:**

"V$RSRCMGRMETRIC"

# V$RULE

V$RULE displays rule statistics. This view has a row for every rule loaded into shared memory.

| Column | Datatype | Description |
|---|---|---|
| RULE_SET_OBJECT_ID | NUMBER | Rule set object ID |
| EVALUATION_CONTEXT_OBJECT_ID | NUMBER | Evaluation context object ID |
| RULE_OWNER | VARCHAR2(30) | Owner of the rule |
| RULE_NAME | VARCHAR2(30) | Name of the rule |
| RULE_CONDITION | VARCHAR2(200) | Rule condition |
| TRUE_HITS | NUMBER | Number of times the rule evaluated to TRUE |
| MAYBE_HITS | NUMBER | Number of times the rule evaluated to MAYBE |
| SQL_EVALUATIONS | NUMBER | Number of evaluations of the rule that were performed by issuing SQL |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$RULE_SET

V$RULE_SET displays rule set statistics. This view has a row for every rule set loaded into shared memory.

**Note:**

Querying the V$RULE_SET view may have a negative impact on performance if a database has a large library cache.

| Column | Datatype | Description |
|---|---|---|
| OWNER | VARCHAR2(30) | Owner of the rule set |
| NAME | VARCHAR2(30) | Name of the rule set |
| CPU_TIME | NUMBER | Total CPU time (in hundredths of a second) spent in evaluation of the rule set |
| ELAPSED_TIME | NUMBER | Total elapsed time (in hundredths of a second) spent in evaluation of the rule set |

| Column | Datatype | Description |
|---|---|---|
| FIRST_LOAD_TIME | DATE | First time the current cached copy of the rule set was loaded |
| LAST_LOAD_TIME | DATE | Last time the current cached copy of the rule set was loaded |
| LAST_LOADING_TIME | NUMBER | Total elapsed time (in hundredths of a second) spent to load the rule set the last time it was loaded |
| SHARABLE_MEM | NUMBER | Shared memory (in bytes) used by the rule set |
| RELOADS | NUMBER | Number of times the rule set object was reloaded in shared memory |
| INVALIDATIONS | NUMBER | Number of times the rule set object was invalidated |
| EVALUATIONS | NUMBER | Number of evaluations on the rule set |
| FIRST_HIT_EVALUATIONS | NUMBER | Number of evaluations on the rule set, with `stop_on_first_hit` set to `TRUE` |
| SIMPLE_RULES_ONLY_EVALUATIONS | NUMBER | Number of evaluations on the rule set, with `simple_rules_only` set to `TRUE` |
| SQL_FREE_EVALUATIONS | NUMBER | Number of evaluations on the rule set which did not internally issue SQL to evaluate rules |
| SQL_EXECUTIONS | NUMBER | Total number of SQL statements executed during evaluation of the rule set |
| CONDITIONS_PROCESSED | NUMBER | Total number of fast (indexed) conditions processed during evaluation of the rule set |
| TRUE_RULES | NUMBER | Total number of `TRUE` rules returned during evaluation of the rule set |
| MAYBE_RULES | NUMBER | Total number of `MAYBE` rules returned during evaluation of the rule set |
| VARIABLE_VALUE_FUNCTION_CALLS | NUMBER | Total number of calls made to user-defined functions to retrieve variable values (specified by the `variable_value_function` field in `RE$VARIABLE_TYPE`) made during evaluation of the rule set |
| VARIABLE_METHOD_FUNCTION_CALLS | NUMBER | Total number of calls made to user-defined functions to retrieve variable method values (specified by the `variable_method_function` field in `RE$VARIABLE_TYPE`) made during evaluation of the rule set |
| EVALUATION_FUNCTION_CALLS | NUMBER | Total number of calls made to user-defined evaluation functions (specified as the `evaluation_function` argument to the `DBMS_RULE_ADM.CREATE_EVALUATION_CONTEXT` procedure) made during evaluation of the rule set |
| RESULT_CACHE_HITS | NUMBER | Number of result cache hits across all the sessions evaluating this rule set |
| IS_RESULT_CACHE | VARCHAR2(3) | Indicates whether this rule set result can be cached (`YES`) or not (`NO`) |
| RESULT_CACHE_ELEMENTS | NUMBER | Number of elements cached within the result cache |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_RULE_ADM.CREATE_EVALUATION_CONTEXT procedure

# V$RULE_SET_AGGREGATE_STATS

V$RULE_SET_AGGREGATE_STATS displays statistics aggregated over all evaluations on all rule sets. This view has a row for each type of statistic.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(80) | Name of the statistic:<br>• rule set evaluations (all) - Total number of evaluations on all rule sets<br>• rule set evaluations (first_hit) - Total number of evaluations on rule sets with stop_on_first_hit set to TRUE<br>• rule set evaluations (simple_rules_only) - Total number of evaluations on rule sets with simple_rules_only set to TRUE<br>• rule set evaluations (SQL free) - Total number of evaluations on rule sets which did not internally issue SQL to evaluate rules<br>• rule set evaluation time (CPU) - Total CPU time (in hundredths of a second) spent in evaluations on rule sets<br>• rule set evaluation time (elapsed) - Total elapsed time (in hundredths of a second) spent in evaluations on rule sets<br>• rule set SQL executions - Total number of SQL statements executed during evaluations on rule sets<br>• rule set conditions processed - Total number of fast (indexed) conditions processed during evaluations on rule sets<br>• rule set true rules - Total number of TRUE rules returned during evaluations on rule sets<br>• rule set maybe rules - Total number of MAYBE rules returned during evaluations on rule sets<br>• rule set user function calls (variable value function) - Total number of calls made to user-defined functions to retrieve variable values (specified by the variable_value_function field in RE$VARIABLE_TYPE) made during evaluations on rule sets<br>• rule set user function calls (variable method function) - Total number of calls made to user-defined |

| Column | Datatype | Description |
|---|---|---|
| | | functions to retrieve variable method values (specified by the `variable_method_function` field in `RE $VARIABLE_TYPE`) made during evaluations on rule sets<br>• `rule set user function calls (evaluation function)` - Total number of calls made to user-defined evaluation functions (specified as the `evaluation_function` argument to the `DBMS_RULE_ADM.CREATE_EVALUATION_CONTEXT` procedure) made during evaluations on rule sets |
| `VALUE` | `NUMBER` | Statistic value |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. The `CON_ID` value in this view is always 0. The rows pertain to the entire CDB or to the non-CDB. |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_RULE_ADM.CREATE_EVALUATION_CONTEXT` procedure

# 9

# Dynamic Performance (V$) Views: V$SCHEDULER_RUNNING_JOBS to V$XSTREAM_TRANSACTION

This chapter contains the dynamic performance views V$SCHEDULER_RUNNING_JOBS to V$XSTREAM_TRANSACTION.

## V$SCHEDULER_RUNNING_JOBS

V$SCHEDULER_RUNNING_JOBS displays information about running Scheduler jobs.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Identifier of the session running the Scheduler job |
| SESSION_SERIAL_NUM | NUMBER | Session serial number |
| JOB_ID | NUMBER | ID of the running Scheduler job |
| PADDR | RAW(4 \| 8) | Process address |
| OS_PROCESS_ID | VARCHAR2(12) | Operating system process ID |
| SESSION_STAT_CPU | INTERVAL DAY(2) TO SECOND(3) | CPU statistics for the session |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire multitenant container database (CDB). This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

## V$SECUREFILE_TIMER

V$SECUREFILE_TIMER displays information about time taken (in microseconds) by functions of SecureFiles. These timer values are collected per session.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(50) | Name of the function |
| LAYER_ID | NUMBER | ID of the layer that the function belongs to:<br>• 0 - Entire Subtree |

| Column | Datatype | Description |
|---|---|---|
| | | • 1 - Delta Updates<br>• 2 - Write gather cache<br>• 3 - Deduplication<br>• 4 - Compression & Encryption<br>• 5 - Inode<br>• 6 - Space<br>• 7 - Utilities<br>• 8 - Row-Column Intersection |
| OWNTIME | NUMBER | Total time taken by the function |
| MAXTIME | NUMBER | Maximum time taken by a single call |
| MINTIME | NUMBER | Minimum time taken by a single call |
| INVOCATIONS | NUMBER | Number of times the function was invoked |
| LAYER_NAME | VARCHAR2(50) | Name of the layer to which the function belongs |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SEGMENT_STATISTICS

V$SEGMENT_STATISTICS displays information about segment-level statistics.

| Column | Datatype | Description |
|---|---|---|
| OWNER | VARCHAR2(128) | Owner of the object |
| OBJECT_NAME | VARCHAR2(128) | Name of the object |
| SUBOBJECT_NAME | VARCHAR2(128) | Name of the subobject |
| TABLESPACE_NAME | VARCHAR2(30) | Name of the table space to which the object belongs |
| TS# | NUMBER | Tablespace number |
| OBJ# | NUMBER | Dictionary object number of the object |
| DATAOBJ# | NUMBER | Data object number of the object |
| OBJECT_TYPE | VARCHAR2(18) | Type of the object |
| STATISTIC_NAME | VARCHAR2(64) | Name of the statistic |
| STATISTIC# | NUMBER | Statistic number |
| VALUE | NUMBER | Statistic value |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SEGSTAT

V$SEGSTAT displays information about segment-level statistics.

| Column | Datatype | Description |
|---|---|---|
| TS# | NUMBER | Tablespace number |
| OBJ# | NUMBER | Dictionary object number |
| DATAOBJ# | NUMBER | Data object number |
| STATISTIC_NAME | VARCHAR2(64) | Name of the statistic |
| STATISTIC# | NUMBER | Statistic number |
| VALUE | NUMBER | Statistic value |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SEGSTAT_NAME

V$SEGSTAT_NAME displays information about segment-level statistics properties.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC# | NUMBER | Statistic number |
| NAME | VARCHAR2(64) | Name of the statistic |
| SAMPLED | VARCHAR2(3) | Indicates whether the statistic was collected by sampling (YES) or not (NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SERV_MOD_ACT_STATS

V$SERV_MOD_ACT_STATS displays the same set of performance statistics as V$SERVICE_STATS except for a specific combination of service/module/action names.

When aggregation is enabled for the service name, module, and action name, then this view provides the timing and work done for calls issued for the business transaction.

| Column | Datatype | Description |
|---|---|---|
| AGGREGATION_TYPE | VARCHAR2(21) | Aggregation statistic type:<br>• SERVICE_MODULE - Action value is NULL and the entry is an aggregate for all actions within a given module<br>• SERVICE_MODULE_ACTION - Action value is NULL only for an empty action, and the aggregation is on the level of service/module/action |
| SERVICE_NAME | VARCHAR2(64) | Service name from V$SERVICES |
| MODULE | VARCHAR2(65) | Module name from DBA_ENABLED_AGGREGATIONS |
| ACTION | VARCHAR2(65) | Action name from DBA_ENABLED_AGGREGATIONS |
| STAT_ID | NUMBER | Statistic identifier |
| STAT_NAME | VARCHAR2(64) | Derived statistic name from V$STATNAME and V$SESS_TIME_MODEL |
| VALUE | NUMBER | Cumulative value (in microseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SERVICE_STATS"
- "V$STATNAME"
- "V$SESS_TIME_MODEL"

# V$SERVICE_EVENT

V$SERVICE_EVENT displays aggregated wait counts and wait times for each wait statistic.

| Column | Datatype | Description |
|---|---|---|
| SERVICE_NAME | VARCHAR2(64) | Service name from V$SERVICES |

| Column | Datatype | Description |
|---|---|---|
| SERVICE_NAME_HASH | NUMBER | Service name hash from V$SERVICES |
| EVENT | VARCHAR2(64) | Name of the wait event; derived statistic name from V$EVENT_NAME |
| EVENT_ID | NUMBER | Identifier of the event |
| TOTAL_WAITS | NUMBER | Total amount of time waited for the event by this service (in hundredths of a second) |
| TOTAL_TIMEOUTS | NUMBER | Total number of timeouts for the event by this service |
| TIME_WAITED | NUMBER | Time waited for the event (in hundredths of a second) |
| AVERAGE_WAIT | NUMBER | Average amount of time waited for the event by this service (in hundredths of a second) |
| MAX_WAIT | NUMBER | Maximum time (in hundredths of a second) waited for the event by this service |
| TIME_WAITED_MICRO | NUMBER | Total time waited for the event (in microseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SERVICE_STATS

V$SERVICE_STATS displays a minimal set of performance statistics. These call rate statistics are used for making run-time routing decisions, for tracking service levels, and for per-instance diagnostics per call rate. The elapsed timing for each call provides a relative value across instances for how well a node is processing SQL calls issued under a service name.

When aggregation is enabled for the Service Name, then this view provides the timing and work done for calls issued for the whole service.

| Column | Datatype | Description |
|---|---|---|
| SERVICE_NAME_HASH | NUMBER | Service name hash from V$SERVICES |
| SERVICE_NAME | VARCHAR2(64) | Service name from V$SERVICES |
| STAT_ID | NUMBER | Statistic identifier |
| STAT_NAME | VARCHAR2(64) | Derived statistic name from V$STATNAME and V$SESS_TIME_MODEL |
| VALUE | NUMBER | Cumulative value (in microseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SERVICE_WAIT_CLASS

V$SERVICE_WAIT_CLASS displays aggregated wait counts and wait times for each wait statistic. An aggregation of these wait classes is used when thresholds are imported.

| Column | Datatype | Description |
|---|---|---|
| SERVICE_NAME | VARCHAR2(64) | Service name from V$SERVICES |
| SERVICE_NAME_HASH | NUMBER | Service name hash from V$SERVICES |
| WAIT_CLASS_ID | NUMBER | Identifier of the wait class |
| WAIT_CLASS# | NUMBER | Number of the wait class |
| WAIT_CLASS | VARCHAR2(64) | Name of the wait class |
| TOTAL_WAITS | NUMBER | Number of times waits of the class occurred for this client |
| TIME_WAITED | NUMBER | Amount of time (in hundredths of a second) spent in the class by this session |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SERVICEMETRIC

V$SERVICEMETRIC displays metric values measured on the most recent time interval period for services executing inside the database. Service metrics are measured in 5-second and 1-minute intervals.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin timestamp for the interval period |
| END_TIME | DATE | End timestamp for the interval period |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | Group ID for the service metric group |
| SERVICE_NAME_HASH | NUMBER | Service name hash |

| Column | Datatype | Description |
|---|---|---|
| SERVICE_NAME | VARCHAR2(64) | Service name |
| CTMHASH | NUMBER | Service create timestamp hash value |
| ELAPSEDPERCALL | NUMBER | Elapsed time per call (in microseconds). This column is deprecated in favor of the DBTIMEPERCALL column. |
| CPUPERCALL | NUMBER | CPU time per call (in microseconds) |
| DBTIMEPERCALL | NUMBER | Elapsed time per call (in microseconds) |
| CALLSPERSEC | NUMBER | Number of user calls per second |
| DBTIMEPERSEC | NUMBER | Database time per second |
| GOODNESS | NUMBER | Indicates how attractive a given instance is with respect to processing the workload that is presented to the service. A lower number is better. This number is internally computed based on the GOAL (LONG or SHORT) that is specified for the particular service. |
| DELTA | NUMBER | Indicates the predicted increase in the goodness for every additional session that is routed to this instance |
| FLAGS | NUMBER | Flags that can be any of the following values:<br>• 0x01 - Service is BLOCKED from accepting new connections<br>• 0x02 - Service is VIOLATING the set threshold on some metric<br>• 0x04 - Goodness is UNKNOWN |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SERVICEMETRIC_HISTORY

V$SERVICEMETRIC_HISTORY displays a recent history of the metric values measured in predefined time interval periods for services executing inside the database. Service metrics are measured in 5-second and 1-minute intervals.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin timestamp for the interval period |
| END_TIME | DATE | End timestamp for the interval period |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | Group ID for the service metric group |
| SERVICE_NAME_HASH | NUMBER | Service name hash |
| SERVICE_NAME | VARCHAR2(64) | Service name |

| Column | Datatype | Description |
|---|---|---|
| CTMHASH | NUMBER | Service create timestamp hash value |
| ELAPSEDPERCALL | NUMBER | Elapsed time per call (in microseconds). This column is deprecated in favor of the DBTIMEPERCALL column. |
| CPUPERCALL | NUMBER | CPU time per call (in microseconds) |
| DBTIMEPERCALL | NUMBER | Elapsed time per call (in microseconds) |
| CALLSPERSEC | NUMBER | Number of user calls per second |
| DBTIMEPERSEC | NUMBER | Database time per second |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SERVICES

V$SERVICES displays information about the services in the database.

| Column | Datatype | Description |
|---|---|---|
| SERVICE_ID | NUMBER | Service ID |
| NAME | VARCHAR2(64) | Name of the service |
| NAME_HASH | NUMBER | Service name hash |
| NETWORK_NAME | VARCHAR2(512) | Network name |
| CREATION_DATE | DATE | Creation date |
| CREATION_DATE_HASH | NUMBER | Creation date hash |
| GOAL | VARCHAR2(12) | Runtime Load Balancing Goal being used to create run-time load balancing and connection load balancing advice:<br>• NONE<br>• SERVICE_TIME - Connections are balanced by response time<br>• THROUGHPUT - Connections are balanced by throughput |
| DTP | VARCHAR2(1) | Indicates whether or not Distributed Transaction Processing is enabled for this service. When Distributed Transaction Processing is set to Y (YES), it means that the service is offered at exactly one instance at a time for XA affinity. Possible values:<br>• Y - YES<br>• N - NO |
| AQ_HA_NOTIFICATION | VARCHAR2(3) | Indicates whether FAN - Fast Application Notification for OCI connections is set (YES) or not (NO) |

| Column | Datatype | Description |
|---|---|---|
| CLB_GOAL | VARCHAR2(5) | Connection load balancing goal used with statistics that are sent to the listeners to determine how new connections are distributed:<br>• LONG - is using session count<br>• SHORT - is using service time or throughput |
| COMMIT_OUTCOME | VARCHAR2(3) | For Transaction Guard, indicates whether the database service associated with the user session has the COMMIT_OUTCOME service attribute enabled (YES) or not (NO).<br>When the attribute is enabled:<br>• The outcome of a COMMIT transaction is durable, which means the status of the last COMMIT executed can be looked up after an outage.<br>• A logical transaction ID (LTXID) is set for each user session at session creation and successful commit.<br>**See Also:** For information about preserving the commit outcome, see *Oracle Database Development Guide*. For information about logical transaction IDs, see *Oracle Database Development Guide* |
| RETENTION_TIME | NUMBER | For Transaction Guard (COMMIT_OUTCOME set to TRUE), this parameter determines the amount of time (in seconds) that the commit outcome is retained in the database. |
| REPLAY_INITIATION_TIMEOUT | NUMBER | For Application Continuity, this option specifies the difference between the time (in seconds) of original execution of the first operation of a request and the time that the replay is ready to start after a successful reconnect. Application Continuity will not replay after the specified amount of time has passed. This option is intended to avoid the unintentional execution of a request when a system is recovered after a long period of time. The default is 5 minutes (300 seconds). |
| SESSION_STATE_CONSISTENCY | VARCHAR2(30) | Describes how non-transactional is changed during a request. This parameter is considered only if failover_type is set to TRANSACTION for Application Continuity. Examples of session state are NLS settings, optimizer preferences, event settings, PL/SQL global variables, temporary tables, advanced queues, LOBs, and result cache. If non-transactional values change after the request starts, the default value of DYNAMIC should be set. Almost all applications should use DYNAMIC mode. If you are unsure, use DYNAMIC mode. |
| GLOBAL | VARCHAR2(3) | Indicates whether the service is global. A global service is managed by Global Service Manager (GSM) and can be provided by multiple databases that contain replicated data. Possible values:<br>• YES: Indicates the service is global<br>• NO: Indicates the service is not global |
| PDB | VARCHAR2(30) | Name of a pluggable database (PDB) associated with a given service. Will contain NULL if a database is a non-CDB or if the service is not associated with a PDB (that is, connecting to a CDB using this service will cause a user to connect to the root.) |
| SQL_TRANSLATION_PROFILE | VARCHAR2(65) | A non-NULL value specifies the initial SQL translation profile for subsequent database connections that use the service and do not specify a SQL translation profile. A NULL value has no effect. |

| Column | Datatype | Description |
|---|---|---|
| MAX_LAG_TIME | VARCHAR2(30) | The maximum replication lag (in seconds) that is acceptable for a data replica to be used for providing the database service. Can only be specified for global services. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SES_OPTIMIZER_ENV

V$SES_OPTIMIZER_ENV displays the contents of the optimizer environment used by each session. When a new session is first created, it automatically inherits its optimizer environment from the optimizer environment defined at the instance level by V$SYS_OPTIMIZER_ENV. The value of certain parameters can be dynamically modified by issuing an ALTER SESSION statement.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier. This column can be used to join with V$SESSION on the SID column. |
| ID | NUMBER | Unique identifier of the parameter in the optimizer environment |
| NAME | VARCHAR2(40) | Name of the parameter |
| SQL_FEATURE | VARCHAR2(64) | Associated feature control ID |
| ISDEFAULT | VARCHAR2(3) | Indicates whether the parameter is set to the default value (YES) or not (NO) |
| VALUE | VARCHAR2(25) | Value of the parameter for the session |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYS_OPTIMIZER_ENV"

# V$SESS_IO

V$SESS_IO displays I/O statistics for each user session.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |
| BLOCK_GETS | NUMBER | Block gets for this session |
| CONSISTENT_GETS | NUMBER | Consistent gets for this session |
| PHYSICAL_READS | NUMBER | Physical reads for this session |
| BLOCK_CHANGES | NUMBER | Block changes for this session |
| CONSISTENT_CHANGES | NUMBER | Consistent changes for this session |
| OPTIMIZED_PHYSICAL_READS | NUMBER | Number of physical reads from Database Smart Flash Cache for this session |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESS_TIME_MODEL

V$SESS_TIME_MODEL displays the session-accumulated time for various operations. The time reported is the total elapsed or CPU time (in microseconds). Any timed operation will buffer at most 5 seconds of time data. Specifically, this means that if a timed operation (such as SQL execution) takes a long period of time to perform, the data published to this view is at most missing 5 seconds of the time accumulated for the operation.

The time values are 8-byte integers and can therefore hold approximately 580,000 years of time before wrapping. Background process time is not included in a statistic value unless the statistic is specifically for background processes.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID (same as in V$SESSION) |
| STAT_ID | NUMBER | Statistic identifier for the time statistic |
| STAT_NAME | VARCHAR2(64) | Name of the statistic (see Table 9-1) |
| VALUE | NUMBER | Amount of time (in microseconds) that the session has spent in this operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

***Table 9-1 V$SESS_TIME_MODEL and V$SYS_TIME_MODEL Statistics***

| Statistic Name | Description |
|---|---|
| DB Time | Amount of elapsed time (in microseconds) spent performing Database user-level calls. This does not include the elapsed time spent on instance background processes such as PMON. |
| DB CPU | Amount of CPU time (in microseconds) spent on database user-level calls. This does not include the CPU time spent on instance background processes such as PMON. |
| background elapsed time | Amount of elapsed time (in microseconds) consumed by database background processes. |
| background CPU time | Amount of CPU time (in microseconds) consumed by database background processes. |
| sequence load elapsed time | Amount of elapsed time spent getting the next sequence number from the data dictionary. If a sequence is cached, then this is the amount of time spent replenishing the cache when it runs out. No time is charged when a sequence number is found in the cache. For non-cached sequences, some time will be charged for every nextval call. |
| parse time elapsed | Amount of elapsed time spent parsing SQL statements. It includes both soft and hard parse time. |
| hard parse elapsed time | Amount of elapsed time spent hard parsing SQL statements. |
| SQL execute elapsed time | Amount of elapsed time SQL statements are executing. Note that for select statements this also includes the amount of time spent performing fetches of query results. |
| connection management call elapsed time | Amount of elapsed time spent performing session connect and disconnect calls. |
| failed parse elapsed time | Amount of time spent performing SQL parses which ultimately fail with some parse error. |
| failed parse (out of shared memory) elapsed time | Amount of time spent performing SQL parses which ultimately fail with error `ORA-04031`. |
| hard parse (sharing criteria) elapsed time | Amount of elapsed time spent performing SQL hard parses when the hard parse resulted from not being able to share an existing cursor in the SQL cache. |
| hard parse (bind mismatch) elapsed time | Amount of elapsed time spent performing SQL hard parses when the hard parse resulted from bind type or bind size mismatch with an existing cursor in the SQL cache. |
| PL/SQL execution elapsed time | Amount of elapsed time spent running the PL/SQL interpreter. This does not include time spent recursively executing/parsing SQL statements or time spent recursively executing the Java VM. |
| PL/SQL compilation elapsed time | Amount of elapsed time spent running the PL/SQL compiler. |
| inbound PL/SQL rpc elapsed time | Time inbound PL/SQL remote procedure calls have spent executing. It includes all time spent recursively executing SQL and JAVA, and therefore is not easily related to "PL/SQL execution elapsed time". |

| Statistic Name | Description |
|---|---|
| Java execution elapsed time | Amount of elapsed time spent running the Java VM. This does not include time spent recursively executing/parsing SQL statements or time spent recursively executing PL/SQL. |
| RMAN cpu time (backup/ restore) | Amount of CPU time (in microseconds) spent in RMAN backup and restore operations. |
| repeated bind elapsed time | Amount of elapsed time spent giving new values to bind variables (rebinding). |
| OLAP engine elapsed time | Amount of time spent performing OLAP session transactions. This includes time spent on database user-level calls, SQL statement execution, and PL/SQL execution within the OLAP transaction. |
| OLAP engine CPU time | Amount of CPU time spent on OLAP session transactions. This includes time spent on database user-level calls, SQL statement execution, and PL/SQL execution within the OLAP transaction. |

The relationships between the statistics listed in Table 9-1 form two trees in which all the time reported by a child in the tree is contained within the parent in the tree. The following are the relationship trees; the number is the level in the given tree.

```
1) background elapsed time
    2) background cpu time
          3) RMAN cpu time (backup/restore)
1) DB time
    2) DB CPU
    2) connection management call elapsed time
    2) sequence load elapsed time
    2) sql execute elapsed time
    2) parse time elapsed
          3) hard parse elapsed time
                4) hard parse (sharing criteria) elapsed time
                    5) hard parse (bind mismatch) elapsed time
          3) failed parse elapsed time
                4) failed parse (out of shared memory) elapsed time
    2) PL/SQL execution elapsed time
    2) inbound PL/SQL rpc elapsed time
    2) PL/SQL compilation elapsed time
    2) Java execution elapsed time
    2) repeated bind elapsed time
```

The relationship between a parent and a child in the tree indicates containment only. Keep the following in mind regarding the tree:

- Children do not necessarily add up to the parent.
- Children are not necessarily exclusive (that is, they may overlap).
- The union of children does not necessarily cover the whole of the parent.

**See Also:**

"V$SYS_TIME_MODEL"

# V$SESSION

V$SESSION displays session information for each current session.

| Column | Datatype | Description |
|---|---|---|
| SADDR | RAW(4 \| 8) | Session address |
| SID | NUMBER | Session identifier |
| SERIAL# | NUMBER | Session serial number. Used to uniquely identify a session's objects. Guarantees that session-level commands are applied to the correct session objects if the session ends and another session begins with the same session ID. |
| AUDSID | NUMBER | Auditing session ID |
| PADDR | RAW(4 \| 8) | Address of the process that owns the session |
| USER# | NUMBER | Oracle user identifier |
| USERNAME | VARCHAR2(30) | Oracle username |
| COMMAND | NUMBER | Command in progress (last statement parsed).<br>You can find the command name for any value *n* returned in this COMMAND column by running this SQL query:<br>`SELECT command_name`<br>`FROM v$sqlcommand`<br>`WHERE command_type = n;`<br>A value of 0 in this COMMAND column means the command is not recorded in V$SESSION. |
| OWNERID | NUMBER | Identifier of the user who owns the migratable session; the column contents are invalid if the value is 2147483644<br>For operations using Parallel Slaves, interpret this value as a 4-byte value. The low-order 2 bytes represent the session number and the high-order bytes represent the instance ID of the query coordinator. |
| TADDR | VARCHAR2(16) | Address of the transaction state object |
| LOCKWAIT | VARCHAR2(16) | Address of the lock the session is waiting for; NULL if none |
| STATUS | VARCHAR2(8) | Status of the session:<br>• ACTIVE - Session currently executing SQL<br>• INACTIVE - Session which is inactive and either has no configured limits or has not yet exceeded the configured limits<br>• KILLED - Session marked to be killed<br>• CACHED - Session temporarily cached for use by Oracle*XA<br>• SNIPED - An inactive session that has exceeded some configured limits (for example, resource limits specified for the resource manager consumer group or idle_time specified in the user's profile). Such sessions will not be allowed to become active again. |
| SERVER | VARCHAR2(9) | Server type: |

| Column | Datatype | Description |
|---|---|---|
| | | • DEDICATED<br>• SHARED<br>• PSEUDO<br>• POOLED<br>• NONE |
| SCHEMA# | NUMBER | Schema user identifier |
| SCHEMANAME | VARCHAR2(30) | Schema user name |
| OSUSER | VARCHAR2(30) | Operating system client user name |
| PROCESS | VARCHAR2(24) | Operating system client process ID |
| MACHINE | VARCHAR2(64) | Operating system machine name |
| PORT | NUMBER | Client port number |
| TERMINAL | VARCHAR2(30) | Operating system terminal name |
| PROGRAM | VARCHAR2(48) | Operating system program name |
| TYPE | VARCHAR2(10) | Session type |
| SQL_ADDRESS | RAW(4 \| 8) | Used with SQL_HASH_VALUE to identify the SQL statement that is currently being executed |
| SQL_HASH_VALUE | NUMBER | Used with SQL_ADDRESS to identify the SQL statement that is currently being executed |
| SQL_ID | VARCHAR2(13) | SQL identifier of the SQL statement that is currently being executed |
| SQL_CHILD_NUMBER | NUMBER | Child number of the SQL statement that is currently being executed |
| SQL_EXEC_START | DATE | Time when the execution of the SQL currently executed by this session started; NULL if SQL_ID is NULL |
| SQL_EXEC_ID | NUMBER | SQL execution identifier; NULL if SQL_ID is NULL or if the execution of that SQL has not yet started (see V$SQL_MONITOR) |
| PREV_SQL_ADDR | RAW(4 \| 8) | Used with PREV_HASH_VALUE to identify the last SQL statement executed |
| PREV_HASH_VALUE | NUMBER | Used with SQL_HASH_VALUE to identify the last SQL statement executed |
| PREV_SQL_ID | VARCHAR2(13) | SQL identifier of the last SQL statement executed |
| PREV_CHILD_NUMBER | NUMBER | Child number of the last SQL statement executed |
| PREV_EXEC_START | DATE | SQL execution start of the last executed SQL statement |
| PREV_EXEC_ID | NUMBER | SQL execution identifier of the last executed SQL statement |
| PLSQL_ENTRY_OBJECT_ID | NUMBER | Object ID of the top-most PL/SQL subprogram on the stack; NULL if there is no PL/SQL subprogram on the stack |

| Column | Datatype | Description |
|---|---|---|
| PLSQL_ENTRY_SUBPROGRAM_ID | NUMBER | Subprogram ID of the top-most PL/SQL subprogram on the stack; NULL if there is no PL/SQL subprogram on the stack |
| PLSQL_OBJECT_ID | NUMBER | Object ID of the currently executing PL/SQL subprogram; NULL if executing SQL |
| PLSQL_SUBPROGRAM_ID | NUMBER | Subprogram ID of the currently executing PL/SQL object; NULL if executing SQL |
| MODULE | VARCHAR2(64) | Name of the currently executing module as set by calling the DBMS_APPLICATION_INFO.SET_MODULE procedure |
| MODULE_HASH | NUMBER | Hash value of the MODULE column |
| ACTION | VARCHAR2(64) | Name of the currently executing action as set by calling the DBMS_APPLICATION_INFO.SET_ACTION procedure |
| ACTION_HASH | NUMBER | Hash value of the ACTION column |
| CLIENT_INFO | VARCHAR2(64) | Information set by the DBMS_APPLICATION_INFO.SET_CLIENT_INFO procedure |
| FIXED_TABLE_SEQUENCE | NUMBER | This contains a number that increases every time the session completes a call to the database and there has been an intervening select from a dynamic performance table. This column can be used by performance monitors to monitor statistics in the database. Each time the performance monitor looks at the database, it only needs to look at sessions that are currently active or have a higher value in this column than the highest value that the performance monitor saw the last time. All the other sessions have been idle since the last time the performance monitor looked at the database. |
| ROW_WAIT_OBJ# | NUMBER | Object ID for the table containing the row specified in ROW_WAIT_ROW# |
| ROW_WAIT_FILE# | NUMBER | Identifier for the data file containing the row specified in ROW_WAIT_ROW#. This column is valid only if the session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is not -1. |
| ROW_WAIT_BLOCK# | NUMBER | Identifier for the block containing the row specified in ROW_WAIT_ROW#. This column is valid only if the session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is not -1. |
| ROW_WAIT_ROW# | NUMBER | Current row being locked. This column is valid only if the session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is not -1. |
| TOP_LEVEL_CALL# | NUMBER | Oracle top level call number |
| LOGON_TIME | DATE | Time of logon |
| LAST_CALL_ET | NUMBER | If the session STATUS is currently ACTIVE, then the value represents the elapsed time (in seconds) since the session has become active. |

| Column | Datatype | Description |
|---|---|---|
| | | If the session STATUS is currently INACTIVE, then the value represents the elapsed time (in seconds) since the session has become inactive. |
| PDML_ENABLED | VARCHAR2(3) | This column has been replaced by the PDML_STATUS column |
| FAILOVER_TYPE | VARCHAR2(13) | Indicates whether and to what extent transparent application failover (TAF) is enabled for the session:<br>• NONE - Failover is disabled for this session<br>• SESSION - Client can fail over its session following a disconnect<br>• SELECT - Client can fail over queries in progress as well<br>**See Also:**<br>• *Oracle Database Net Services Administrator's Guide* for more information on TAF<br>• *Oracle Database Net Services Administrator's Guide* for information on configuring TAF |
| FAILOVER_METHOD | VARCHAR2(10) | Indicates the transparent application failover method for the session:<br>• NONE - Failover is disabled for this session<br>• BASIC - Client itself reconnects following a disconnect<br>• PRECONNECT - Backup instance can support all connections from every instance for which it is backed up |
| FAILED_OVER | VARCHAR2(3) | Indicates whether the session is running in failover mode and failover has occurred (YES) or not (NO) |
| RESOURCE_CONSUMER_GROUP | VARCHAR2(32) | Name of the session's current resource consumer group |
| PDML_STATUS | VARCHAR2(8) | If ENABLED, the session is in a PARALLEL DML enabled mode. If DISABLED, PARALLEL DML enabled mode is not supported for the session. If FORCED, the session has been altered to force PARALLEL DML. |
| PDDL_STATUS | VARCHAR2(8) | If ENABLED, the session is in a PARALLEL DDL enabled mode. If DISABLED, PARALLEL DDL enabled mode is not supported for the session. If FORCED, the session has been altered to force PARALLEL DDL. |
| PQ_STATUS | VARCHAR2(8) | If ENABLED, the session is in a PARALLEL QUERY enabled mode. If DISABLED, PARALLEL QUERY enabled mode is not supported for the session. If FORCED, the session has been altered to force PARALLEL QUERY. |
| CURRENT_QUEUE_DURATION | NUMBER | If queued (1), the current amount of time the session has been queued. If not currently queued, the value is 0. |
| CLIENT_IDENTIFIER | VARCHAR2(64) | Client identifier of the session |
| BLOCKING_SESSION_STATUS | VARCHAR2(11) | This column provides details on whether there is a blocking session:<br>• VALID - there is a blocking session, and it is identified in the BLOCKING_INSTANCE and BLOCKING_SESSION columns<br>• NO HOLDER - there is no session blocking this session<br>• NOT IN WAIT - this session is not in a wait<br>• UNKNOWN - the blocking session is unknown |

| Column | Datatype | Description |
|---|---|---|
| BLOCKING_INSTANCE | NUMBER | Instance identifier of the blocking session. This column is valid only if BLOCKING_SESSION_STATUS has the value VALID. |
| BLOCKING_SESSION | NUMBER | Session identifier of the blocking session. This column is valid only if BLOCKING_SESSION_STATUS has the value VALID. |
| FINAL_BLOCKING_SESSION_STATUS | VARCHAR2(11) | The final blocking session is the final element in the wait chain constructed by following the sessions that are blocked by one another starting with this session. In the case of a cyclical wait chain, one of the sessions in the wait chain will be chosen as the final blocker.<br>This column provides details on whether there is a final blocking session:<br>• VALID - there is a final blocking session and it is identified in the FINAL_BLOCKING_INSTANCE and FINAL_BLOCKING_SESSION columns<br>• NO HOLDER - there is no session blocking this session<br>• NOT IN WAIT - this session is not in a wait<br>• UNKNOWN - the final blocking session is unknown |
| FINAL_BLOCKING_INSTANCE | NUMBER | Instance identifier of the final blocking session. This column is valid only if FINAL_BLOCKING_SESSION_STATUS has the value VALID. |
| FINAL_BLOCKING_SESSION | NUMBER | Session identifier of the blocking session. This column is valid only if FINAL_BLOCKING_SESSION_STATUS has the value VALID. |
| SEQ# | NUMBER | A number that uniquely identifies the current or last wait (incremented for each wait) |
| EVENT# | NUMBER | Event number |
| EVENT | VARCHAR2(64) | Resource or event for which the session is waiting<br>**See Also:** Oracle Wait Events |
| P1TEXT | VARCHAR2(64) | Description of the first wait event parameter |
| P1 | NUMBER | First wait event parameter (in decimal) |
| P1RAW | RAW(8) | First wait event parameter (in hexadecimal) |
| P2TEXT | VARCHAR2(64) | Description of the second wait event parameter |
| P2 | NUMBER | Second wait event parameter (in decimal) |
| P2RAW | RAW(8) | Second wait event parameter (in hexadecimal)[1] |
| P3TEXT | VARCHAR2(64) | Description of the third wait event parameter |
| P3 | NUMBER | Third wait event parameter (in decimal) |
| P3RAW | RAW(8) | Third wait event parameter (in hexadecimal)[1] |
| WAIT_CLASS_ID | NUMBER | Identifier of the class of the wait event |
| WAIT_CLASS# | NUMBER | Number of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | Name of the class of the wait event |

| Column | Datatype | Description |
|---|---|---|
| WAIT_TIME | NUMBER | If the session is currently waiting, then the value is 0. If the session is not in a wait, then the value is as follows:<br>• > 0 - Value is the duration of the last wait in hundredths of a second<br>• -1 - Duration of the last wait was less than a hundredth of a second<br>• -2 - Parameter TIMED_STATISTICS was set to false<br>This column has been deprecated in favor of the columns WAIT_TIME_MICRO and STATE. |
| SECONDS_IN_WAIT | NUMBER | If the session is currently waiting, then the value is the amount of time waited for the current wait. If the session is not in a wait, then the value is the amount of time since the start of the last wait.<br>This column has been deprecated in favor of the columns WAIT_TIME_MICRO and TIME_SINCE_LAST_WAIT_MICRO. |
| STATE | VARCHAR2(19) | Wait state:<br>• WAITING - Session is currently waiting<br>• WAITED UNKNOWN TIME - Duration of the last wait is unknown; this is the value when the parameter TIMED_STATISTICS is set to false<br>• WAITED SHORT TIME - Last wait was less than a hundredth of a second<br>• WAITED KNOWN TIME - Duration of the last wait is specified in the WAIT_TIME column |
| WAIT_TIME_MICRO | NUMBER | Amount of time waited (in microseconds). If the session is currently waiting, then the value is the time spent in the current wait. If the session is currently not in a wait, then the value is the amount of time waited in the last wait. |
| TIME_REMAINING_MICRO | NUMBER | Value is interpreted as follows:<br>• > 0 - Amount of time remaining for the current wait (in microseconds)<br>• 0 - Current wait has timed out<br>• -1 - Session can indefinitely wait in the current wait<br>• NULL - Session is not currently waiting |
| TIME_SINCE_LAST_WAIT_MICRO | NUMBER | Time elapsed since the end of the last wait (in microseconds). If the session is currently in a wait, then the value is 0. |
| SERVICE_NAME | VARCHAR2(64) | Service name of the session |
| SQL_TRACE | VARCHAR2(8) | Indicates whether SQL tracing is enabled (ENABLED) or disabled (DISABLED) |
| SQL_TRACE_WAITS | VARCHAR2(5) | Indicates whether wait tracing is enabled (TRUE) or not (FALSE) |
| SQL_TRACE_BINDS | VARCHAR2(5) | Indicates whether bind tracing is enabled (TRUE) or not (FALSE) |
| SQL_TRACE_PLAN_STATS | VARCHAR2(10) | Frequency at which row source statistics are dumped in the trace files for each cursor:<br>• never<br>• first_execution<br>• all_executions |

| Column | Datatype | Description |
|---|---|---|
| SESSION_EDITION_ID | NUMBER | Shows the value that, in the session, would be reported by sys_context('USERENV', 'SESSION_EDITION_ID') |
| CREATOR_ADDR | RAW(4 \| 8) | Address of the creating process or circuit |
| CREATOR_SERIAL# | NUMBER | Serial number of the creating process or circuit |
| ECID | VARCHAR2(64) | Execution context identifier (sent by Application Server) |
| SQL_TRANSLATION_PROFILE_ID | NUMBER | Object number of the SQL translation profile |
| PGA_TUNABLE_MEM | NUMBER | The amount of tunable PGA memory (in bytes).<br>Untunable memory is PGA_ALLOC_MEM from V$PROCESS minus PGA_TUNABLE_MEM from V$SESSION. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| EXTERNAL_NAME | VARCHAR2(1024) | External name of the database user. For enterprise users, returns the Oracle Internet Directory DN. |

[1] The P1RAW, P2RAW, and P3RAW columns display the same values as the P1, P2, and P3 columns, except that the numbers are displayed in hexadecimal.

**See Also:**

- *Oracle Database Performance Tuning Guide* for an example of using V$SESSION to help identify an object that is waiting for buffer busy waits
- *Oracle Database Performance Tuning Guide* for an example of using V$SESSION to determine which sessions are waiting for I/O
- *Oracle Database PL/SQL Packages and Types Reference* for more information on the DBMS_APPLICATION_INFO package

# V$SESSION_BLOCKERS

V$SESSION_BLOCKERS displays the blocker sessions for each blocked session. Each row represents a blocked and blocker session pair. If a session is blocked by multiple sessions there will be multiple rows for that blocked session. The maximum number of blocker sessions displayed for a single blocked session is 30. If a session is not blocked by other sessions, then there will be no row in this view for that session.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Blocked session's Oracle session identifier |
| SESS_SERIAL# | NUMBER | Blocked session's Oracle session serial number |

| Column | Datatype | Description |
|---|---|---|
| WAIT_ID | NUMBER | A number identifying the wait for the blocked session |
| WAIT_EVENT | NUMBER | Resource or event number for which the blocked session is waiting |
| WAIT_EVENT_TEXT | VARCHAR2(64) | Resource or event for which the blocked session is waiting |
| BLOCKER_INSTANCE_ID | NUMBER | Blocker session's instance identifier |
| BLOCKER_SID | NUMBER | Blocker session's Oracle session identifier |
| BLOCKER_SESS_SERIAL# | NUMBER | Blocker session's Oracle session serial number |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESSION_CONNECT_INFO

V$SESSION_CONNECT_INFO displays information about network connections for all currently logged in sessions.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier (can be used to join this view with V$SESSION) |
| SERIAL# | NUMBER | Session serial number. Used to uniquely identify a session's objects. Guarantees that session-level commands are applied to the correct session objects if the session ends and another session begins with the same session ID. (Can be used to join this view with V$SESSION.) |
| AUTHENTICATION_TYPE | VARCHAR2(26) | How the user was authenticated:<br>• DATABASE - Username/password authentication<br>• OS - Operating system external user authentication<br>• NETWORK - Network protocol or ANO authentication<br>• PROXY - OCI proxy connection authentication<br>• SERVER<br>• PASSWORD<br>• EXTERNAL ADAPTERS<br>• INTERNAL<br>• GLOBAL<br>• EXTERNAL<br>• PASSWORD BASED GLOBAL USER |
| OSUSER | VARCHAR2(30) | External username for this database user |
| NETWORK_SERVICE_BANNER | VARCHAR2(4000) | Product banners for each Oracle Net service used for this connection (one row per banner) |

| Column | Datatype | Description |
|---|---|---|
| CLIENT_CHARSET | VARCHAR2(40) | Client character set as specified in the NLS_LANG environment variable or in the OCIEnvNlsCreate() call; Unknown if the Oracle client is older than release 11.1 or the connection is through the JDBC thin driver |
| CLIENT_CONNECTION | VARCHAR2(13) | Client server connection flags:<br>• Heterogeneous<br>• Homogeneous |
| CLIENT_OCI_LIBRARY | VARCHAR2(27) | OCI client library:<br>• Home-based<br>• Full Instant Client<br>• XE Instant Client<br>• Light Weight Instant Client<br>• OCI<br>• Unknown |
| CLIENT_VERSION | VARCHAR2(40) | Client library version number |
| CLIENT_DRIVER | VARCHAR2(9) | Client driver name |
| CLIENT_LOBATTR | VARCHAR2(23) | Client LOB flags:<br>• Client Temp Lob Rfc On<br>• Client Temp Lob Rfc Off |
| CLIENT_REGID | NUMBER | Query cache registration ID sent by the client |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SESSION"

# V$SESSION_CURSOR_CACHE

V$SESSION_CURSOR_CACHE displays information on cursor usage for the current session.

**Note:**

The V$SESSION_CURSOR_CACHE view is not a measure of the effectiveness of the SESSION_CACHED_CURSORS initialization parameter.

| Column | Datatype | Description |
|---|---|---|
| MAXIMUM | NUMBER | Maximum number of cursors to cache. Once you hit this number, some cursors will need to be closed in order to open more. The value in this column is derived from the initialization parameter SESSION_CACHED_CURSORS. |
| COUNT | NUMBER | Current number of cursors (whether they are in use or not) |
| OPENS | NUMBER | Cumulative total of cursor opens minus one. This is because the cursor that is currently open and being used for this query is not counted in the OPENS statistic. |
| HITS | NUMBER | Cumulative total of cursor open hits |
| HIT_RATIO | NUMBER | Ratio of the number of times an open cursor was found divided by the number of times a cursor was sought |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"SESSION_CACHED_CURSORS"

# V$SESSION_EVENT

V$SESSION_EVENT displays information on waits for an event by a session. Note that the TIME_WAITED and AVERAGE_WAIT columns will contain a value of zero on those platforms that do not support a fast timing mechanism. If you are running on one of these platforms and you want this column to reflect true wait times, then you must set TIMED_STATISTICS to true in the parameter file; doing this will have a small negative effect on system performance.

**See Also:**

"TIMED_STATISTICS"

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | ID of the session |
| EVENT | VARCHAR2(64) | Name of the wait event<br>**See Also:** Oracle Wait Events |
| TOTAL_WAITS | NUMBER | Total number of waits for the event by the session |
| TOTAL_TIMEOUTS | NUMBER | Total number of timeouts for the event by the session |

| Column | Datatype | Description |
|---|---|---|
| TIME_WAITED | NUMBER | Total amount of time waited for the event by the session (in hundredths of a second) |
| AVERAGE_WAIT | NUMBER | Average amount of time waited for the event by the session (in hundredths of a second) |
| MAX_WAIT | NUMBER | Maximum time waited for the event by the session (in hundredths of a second) |
| TIME_WAITED_MICRO | NUMBER | Total amount of time waited for the event by the session (in microseconds) |
| EVENT_ID | NUMBER | Identifier of the wait event |
| WAIT_CLASS_ID | NUMBER | Identifier of the class of the wait event |
| WAIT_CLASS# | NUMBER | Number of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | Name of the class of the wait event |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESSION_FIX_CONTROL

V$SESSION_FIX_CONTROL displays information about Fix Control (enabled/disabled) for the current session.

| Column | Datatype | Description |
|---|---|---|
| SESSION_ID | NUMBER | Session identifier (can be used to join this view with V$SESSION) |
| BUGNO | NUMBER | Bug number (as fix control identifier) |
| VALUE | NUMBER | Current value set for the fix control |
| SQL_FEATURE | VARCHAR2(64) | Feature control ID |
| DESCRIPTION | VARCHAR2(64) | Description of the fix control |
| OPTIMIZER_FEATURE_ENABLE | VARCHAR2(25) | Version on (and after) which the fix is enabled by default |
| EVENT | NUMBER | Event formerly used to control the fix |
| IS_DEFAULT | NUMBER | Indicates whether the current value is the same as the default (1) or not (0) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESSION_LONGOPS

V$SESSION_LONGOPS displays the status of various operations that run for longer than 6 seconds (in absolute time). These operations currently include many backup and recovery functions, statistics gathering, and query execution, and more operations are added for every Oracle release.

To monitor query execution progress, you must be using the cost-based optimizer and you must:

- Set the TIMED_STATISTICS or SQL_TRACE parameters to true
- Gather statistics for your objects with the DBMS_STATS package

You can add information to this view about application-specific long-running operations by using the DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS procedure.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Identifier of the session processing the long-running operation. If multiple sessions are cooperating in the long-running operation, then SID corresponds to the main or master session. |
| SERIAL# | NUMBER | Serial number of the session processing the long-running operation. If multiple sessions are cooperating in the long-running operation, then SERIAL# corresponds to the main or master session. SERIAL# is used to uniquely identify a session's objects. Guarantees that session-level commands are applied to the correct session objects if the session ends and another session begins with the same session ID. |
| OPNAME | VARCHAR2(64) | Brief description of the operation |
| TARGET | VARCHAR2(64) | Object on which the operation is carried out |
| TARGET_DESC | VARCHAR2(32) | Description of the target |
| SOFAR | NUMBER | Units of work done so far for the operation specified in the OPNAME column |
| TOTALWORK | NUMBER | Total units of work for the operation specified in the OPNAME column |
| UNITS | VARCHAR2(32) | Units of measurement |
| START_TIME | DATE | Starting time of the operation |
| LAST_UPDATE_TIME | DATE | Time when statistics were last updated for the operation |
| TIMESTAMP | DATE | Timestamp specific to the operation |
| TIME_REMAINING | NUMBER | Estimate (in seconds) of time remaining for the operation to complete |

| Column | Datatype | Description |
|---|---|---|
| ELAPSED_SECONDS | NUMBER | Number of elapsed seconds from the start of the operations |
| CONTEXT | NUMBER | Context |
| MESSAGE | VARCHAR2(512) | Statistics summary message |
| USERNAME | VARCHAR2(30) | User ID of the user performing the operation |
| SQL_ADDRESS | RAW(4 \| 8) | Used with the value of the SQL_HASH_VALUE column to identify the SQL statement associated with the operation |
| SQL_HASH_VALUE | NUMBER | Used with the value of the SQL_ADDRESS column to identify the SQL statement associated with the operation |
| SQL_ID | VARCHAR2(13) | SQL identifier of the SQL statement associated with the long operation, if any |
| SQL_PLAN_HASH_VALUE | NUMBER | SQL plan hash value; NULL if SQL_ID is NULL |
| SQL_EXEC_START | DATE | Time when the execution of the SQL started; NULL if SQL_ID is NULL |
| SQL_EXEC_ID | NUMBER | SQL execution identifier (see V$SQL_MONITOR) |
| SQL_PLAN_LINE_ID | NUMBER | SQL plan line ID corresponding to the long operation; NULL if the long operation is not associated with a line of the execution plan |
| SQL_PLAN_OPERATION | VARCHAR2(30) | Plan operation name; NULL if SQL_PLAN_LINE_ID is NULL |
| SQL_PLAN_OPTIONS | VARCHAR2(30) | Plan operation options; NULL if SQL_PLAN_LINE_ID is NULL |
| QCSID | NUMBER | Session identifier of the parallel coordinator |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "TIMED_STATISTICS"
- "SQL_TRACE"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STATS package

# V$SESSION_OBJECT_CACHE

V$SESSION_OBJECT_CACHE displays object cache statistics for the current user session on the local server (instance).

| Column | Datatype | Description |
|---|---|---|
| PINS | NUMBER | Number of object pins or look-ups in the cache |
| HITS | NUMBER | Number of object pins that found the object already in the cache |
| TRUE_HITS | NUMBER | Number of object pins that found the object already in the cache and in the desired state (thus, not requiring refresh from the database) |
| HIT_RATIO | NUMBER | Ratio of HITS / PINS |
| TRUE_HIT_RATIO | NUMBER | Ratio of TRUE_HITS/PINS |
| OBJECT_REFRESHES | NUMBER | Number of objects in the cache that were refreshed with a new value from the database |
| CACHE_REFRESHES | NUMBER | Number of times the whole cache (all objects) were refreshed |
| OBJECT_FLUSHES | NUMBER | Number of objects in the cache that were flushed to the database |
| CACHE_FLUSHES | NUMBER | Number of times the whole cache (all objects) were flushed to the database |
| CACHE_SHRINKS | NUMBER | Number of times the cache was shrunk to the optimal size |
| CACHED_OBJECTS | NUMBER | Number of objects currently cached |
| PINNED_OBJECTS | NUMBER | Number of objects currently pinned |
| CACHE_SIZE | NUMBER | Current size of the cache (in bytes) |
| OPTIMAL_SIZE | NUMBER | Optimal size of the cache (in bytes) |
| MAXIMUM_SIZE | NUMBER | Maximum size of the cache (in bytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESSION_WAIT

V$SESSION_WAIT displays the current or last wait for each session.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier; maps to V$SESSION.SID |

| Column | Datatype | Description |
|---|---|---|
| SEQ# | NUMBER | A number that uniquely identifies the current or last wait (incremented for each wait) |
| EVENT | VARCHAR2(64) | Resource or event for which the session is waiting<br>**See Also:** Oracle Wait Events |
| P1TEXT | VARCHAR2(64) | Description of the first wait event parameter |
| P1 | NUMBER | First wait event parameter (in decimal) |
| P1RAW | RAW(8) | First wait event parameter (in hexadecimal) |
| P2TEXT | VARCHAR2(64) | Description of the second wait event parameter |
| P2 | NUMBER | Second wait event parameter (in decimal) |
| P2RAW | RAW(8) | Second wait event parameter (in hexadecimal)[1] |
| P3TEXT | VARCHAR2(64) | Description of the third wait event parameter |
| P3 | NUMBER | Third wait event parameter (in decimal) |
| P3RAW | RAW(8) | Third wait event parameter (in hexadecimal)[1] |
| WAIT_CLASS_ID | NUMBER | Identifier of the class of the wait event |
| WAIT_CLASS# | NUMBER | Number of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | Name of the class of the wait event |
| WAIT_TIME | NUMBER | If the session is currently waiting, then the value is 0. If the session is not in a wait, then the value is as follows:<br>• > 0 - Value is the duration of the last wait in hundredths of a second<br>• -1 - Duration of the last wait was less than a hundredth of a second<br>• -2 - Parameter TIMED_STATISTICS was set to false<br>This column has been deprecated in favor of the columns WAIT_TIME_MICRO and STATE. |
| SECONDS_IN_WAIT | NUMBER | If the session is currently waiting, then the value is the amount of time waited for the current wait. If the session is not in a wait, then the value is the amount of time since the start of the last wait.<br>This column has been deprecated in favor of the columns WAIT_TIME_MICRO and TIME_SINCE_LAST_WAIT_MICRO. |
| STATE | VARCHAR2(19) | Wait state:<br>• WAITING - Session is currently waiting<br>• WAITED UNKNOWN TIME - Duration of the last wait is unknown; this is the value when the parameter TIMED_STATISTICS is set to false<br>• WAITED SHORT TIME - Last wait was less than a hundredth of a second<br>• WAITED KNOWN TIME - Duration of the last wait is specified in the WAIT_TIME column |

| Column | Datatype | Description |
|---|---|---|
| WAIT_TIME_MICRO | NUMBER | Amount of time waited (in microseconds). If the session is currently waiting, then the value is the time spent in the current wait. If the session is currently not in a wait, then the value is the amount of time waited in the last wait. |
| TIME_REMAINING_MICRO | NUMBER | Value is interpreted as follows:<br>• > 0 - Amount of time remaining for the current wait (in microseconds)<br>• 0 - Current wait has timed out<br>• -1 - Session can indefinitely wait in the current wait<br>• NULL - Session is not currently waiting |
| TIME_SINCE_LAST_WAIT_MICRO | NUMBER | Time elapsed since the end of the last wait (in microseconds). If the session is currently in a wait, then the value is 0. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] The P1RAW, P2RAW, and P3RAW columns display the same values as the P1, P2, and P3 columns, except that the numbers are displayed in hexadecimal.

**See Also:**

"TIMED_STATISTICS" and Oracle Wait Events

# V$SESSION_WAIT_CLASS

V$SESSION_WAIT_CLASS displays the time spent in various wait event operations on a per-session basis.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |
| SERIAL# | NUMBER | Serial number |
| WAIT_CLASS_ID | NUMBER | Identifier of the wait class |
| WAIT_CLASS# | NUMBER | Number of the wait class |
| WAIT_CLASS | VARCHAR2(64) | Name of the wait class |
| TOTAL_WAITS | NUMBER | Number of times waits of the class occurred for the session |
| TIME_WAITED | NUMBER | Amount of time spent in the wait class by the session |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESSION_WAIT_HISTORY

V$SESSION_WAIT_HISTORY displays the last 10 wait events for each active session.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |
| SEQ# | NUMBER | Sequence of wait events; 1 is the most recent |
| EVENT# | NUMBER | Event number |
| EVENT | VARCHAR2(64) | Resource or event for which the session is waiting |
| P1TEXT | VARCHAR2(64) | Description of the first wait event parameter |
| P1 | NUMBER | First wait event parameter (in decimal) |
| P2TEXT | VARCHAR2(64) | Description of the second wait event parameter |
| P2 | NUMBER | Second wait event parameter (in decimal) |
| P3TEXT | VARCHAR2(64) | Description of the third wait event parameter |
| P3 | NUMBER | Third wait event parameter (in decimal) |
| WAIT_TIME | NUMBER | Amount of time waited (in hundredths of a second) |
| WAIT_TIME_MICRO | NUMBER | Amount of time waited (in microseconds) |
| TIME_SINCE_LAST_WAIT_MICRO | NUMBER | Time elapsed (in microseconds) since the end of the previous wait in the wait history |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESSIONS_COUNT

V$SESSIONS_COUNT displays the current number of sessions for each PDB.

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CURRENT_COUNT | NUMBER | Displays the current number of sessions for the PDB. This count includes recursive sessions. |

# V$SESSMETRIC

V$SESSMETRIC displays the metric values for all sessions.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| SESSION_ID | NUMBER | Session ID |
| SESSION_SERIAL_NUM | NUMBER | Session serial number |
| CPU | NUMBER | CPU usage |
| PHYSICAL_READS | NUMBER | Number of physical reads |
| LOGICAL_READS | NUMBER | Number of logical reads |
| PGA_MEMORY | NUMBER | PGA size at the end of the interval |
| HARD_PARSES | NUMBER | Number of hard parses |
| SOFT_PARSES | NUMBER | Number of soft parses |
| PHYSICAL_READ_PCT | NUMBER | Physical read ratio |
| LOGICAL_READ_PCT | NUMBER | Logical read ratio |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SESSTAT

V$SESSTAT displays user session statistics. To find the name of the statistic associated with each statistic number (STATISTIC#), query the V$STATNAME view.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier |

| Column | Datatype | Description |
|---|---|---|
| STATISTIC# | NUMBER | Statistic number<br>**Note:** Statistics numbers are not guaranteed to remain constant from one release to another. Therefore, you should rely on the statistics name rather than its number in your applications. |
| VALUE | NUMBER | Statistic value |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$STATNAME" and Statistics Descriptions

# V$SGA

V$SGA displays summary information about the system global area (SGA).

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(20) | SGA component group |
| VALUE | NUMBER | Memory size (in bytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SGA_CURRENT_RESIZE_OPS

V$SGA_CURRENT_RESIZE_OPS displays information about SGA resize operations which are currently in progress. An operation can be a grow or a shrink of a dynamic SGA component. All sizes are expressed in bytes.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT | VARCHAR2(64) | Component name |
| OPER_TYPE | VARCHAR2(13) | Operation type:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW |

| Column | Datatype | Description |
|---|---|---|
| | | • SHRINK<br>• SHRINK_CANCEL |
| OPER_MODE | VARCHAR2(9) | Operation mode:<br>• MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| PARAMETER | VARCHAR2(80) | Name of the parameter for the resize operation |
| INITIAL_SIZE | NUMBER | Parameter value at the start of the operation |
| TARGET_SIZE | NUMBER | Desired value of the parameter after the resize |
| CURRENT_SIZE | NUMBER | Current value of the parameter |
| START_TIME | DATE | Start time of the operation |
| LAST_UPDATE_TIME | DATE | Last time progress was made for the operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SGA_DYNAMIC_COMPONENTS

V$SGA_DYNAMIC_COMPONENTS displays information about the dynamic SGA components. This view summarizes information based on all completed SGA resize operations since instance startup. All sizes are expressed in bytes.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT | VARCHAR2(64) | Component name |
| CURRENT_SIZE | NUMBER | Current size of the component |
| MIN_SIZE | NUMBER | Minimum size of the component since instance startup |
| MAX_SIZE | NUMBER | Maximum size of the component since instance startup |
| USER_SPECIFIED_SIZE | NUMBER | Value of the user parameter for the component |
| OPER_COUNT | NUMBER | Number of operations since instance startup |
| LAST_OPER_TYPE | VARCHAR2(13) | Last completed operation for the component:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW<br>• SHRINK<br>• SHRINK_CANCEL |

| Column | Datatype | Description |
|---|---|---|
| LAST_OPER_MODE | VARCHAR2(9) | Mode of the last completed operation:<br>• MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| LAST_OPER_TIME | DATE | Start time of the last completed operation |
| GRANULE_SIZE | NUMBER | Granularity of the grow or the shrink operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SGA_DYNAMIC_FREE_MEMORY

V$SGA_DYNAMIC_FREE_MEMORY displays information about the amount of SGA memory available for future dynamic SGA resize operations.

| Column | Datatype | Description |
|---|---|---|
| CURRENT_SIZE | NUMBER | Amount of available memory (in bytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SGA_RESIZE_OPS

V$SGA_RESIZE_OPS displays information about the last 800 completed SGA resize operations. This does not include in-progress operations. All sizes are expressed in bytes.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT | VARCHAR2(64) | Component name |
| OPER_TYPE | VARCHAR2(13) | Operation type:<br>• STATIC<br>• INITIALIZING<br>• DISABLED<br>• GROW<br>• SHRINK<br>• SHRINK_CANCEL |
| OPER_MODE | VARCHAR2(9) | Operation mode: |

| Column | Datatype | Description |
|---|---|---|
| | | • MANUAL<br>• DEFERRED<br>• IMMEDIATE |
| PARAMETER | VARCHAR2(80) | Name of the parameter for the resize operation |
| INITIAL_SIZE | NUMBER | Parameter value at the start of the operation |
| TARGET_SIZE | NUMBER | Requested value of the parameter after the resize |
| FINAL_SIZE | NUMBER | Real value of the parameter after the resize |
| STATUS | VARCHAR2(9) | Completion status of the operation:<br>• INACTIVE<br>• PENDING<br>• COMPLETE<br>• CANCELLED<br>• ERROR |
| START_TIME | DATE | Start time of the operation |
| END_TIME | DATE | End time of the operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SGA_TARGET_ADVICE

V$SGA_TARGET_ADVICE displays information about the SGA_TARGET initialization parameter.

| Column | Datatype | Description |
|---|---|---|
| SGA_SIZE | NUMBER | Size of the SGA |
| SGA_SIZE_FACTOR | NUMBER | Ratio between the SGA_SIZE and the current size of the SGA |
| ESTD_DB_TIME | NUMBER | Estimated DB_TIME for this SGA_SIZE |
| ESTD_DB_TIME_FACTOR | NUMBER | Ratio between ESTD_DB_TIME and DB_TIME for the current size of the SGA |
| ESTD_PHYSICAL_READS | NUMBER | Estimated number of physical reads |
| ESTD_BUFFER_CACHE_SIZE | NUMBER | Estimated size of the buffer cache |
| ESTD_SHARED_POOL_SIZE | NUMBER | Estimated size of the shared pool |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"SGA_TARGET"

# V$SGAINFO

V$SGAINFO displays size information about the SGA, including the sizes of different SGA items, the granule size, and free memory.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(32) | Name of the SGA size item |
| BYTES | NUMBER | Size of the item (in bytes) |
| RESIZEABLE | VARCHAR2(3) | Indicates whether the item is resizeable (Yes) or not (No) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SGASTAT

V$SGASTAT displays detailed information on the system global area (SGA).

| Column | Datatype | Description |
|---|---|---|
| POOL | VARCHAR2(12) | Designates the pool in which the memory in NAME resides:<br>• shared pool - Memory is allocated from the shared pool<br>• large pool - Memory is allocated from the large pool<br>• java pool - Memory is allocated from the Java pool<br>• streams pool - Memory is allocated from the Streams pool |
| NAME | VARCHAR2(26) | SGA component name |
| BYTES | NUMBER | Memory size (in bytes) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SHARED_POOL_ADVICE

V$SHARED_POOL_ADVICE displays information about estimated parse time in the shared pool for different pool sizes. The sizes range from 10% of the current shared pool size or the amount of pinned library cache memory (whichever is higher) to 200% of the current shared pool size, in equal intervals. The value of the interval depends on the current size of the shared pool.

| Column | Datatype | Description |
|---|---|---|
| SHARED_POOL_SIZE_FOR_ESTIMATE | NUMBER | Shared pool size for the estimate (in megabytes) |
| SHARED_POOL_SIZE_FACTOR | NUMBER | Size factor with respect to the current shared pool size |
| ESTD_LC_SIZE | NUMBER | Estimated memory in use by the library cache (in megabytes) |
| ESTD_LC_MEMORY_OBJECTS | NUMBER | Estimated number of library cache memory objects in the shared pool of the specified size |
| ESTD_LC_TIME_SAVED | NUMBER | Estimated elapsed parse time saved (in seconds), owing to library cache memory objects being found in a shared pool of the specified size. This is the time that would have been spent in reloading the required objects in the shared pool had they been aged out due to insufficient amount of available free memory. |
| ESTD_LC_TIME_SAVED_FACTOR | NUMBER | Estimated parse time saved factor with respect to the current shared pool size |
| ESTD_LC_LOAD_TIME | NUMBER | Estimated elapsed time (in seconds) for parsing in a shared pool of the specified size |
| ESTD_LC_LOAD_TIME_FACTOR | NUMBER | Estimated load time factor with respect to the current shared pool size |
| ESTD_LC_MEMORY_OBJECT_HITS | NUMBER | Estimated number of times a library cache memory object was found in a shared pool of the specified size |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SHARED_POOL_RESERVED

V$SHARED_POOL_RESERVED displays statistics that help you tune the reserved pool and space within the shared pool.

| Column | Datatype | Description |
|---|---|---|
| FREE_SPACE | NUMBER | Total amount of free space on the reserved list |
| AVG_FREE_SIZE | NUMBER | Average size of the free memory on the reserved list[1] |
| FREE_COUNT | NUMBER | Number of free pieces of memory on the reserved list[1] |
| MAX_FREE_SIZE | NUMBER | Size of the largest free piece of memory on the reserved list[1] |
| USED_SPACE | NUMBER | Total amount of used memory on the reserved list[1] |
| AVG_USED_SIZE | NUMBER | Average size of the used memory on the reserved list[1] |
| USED_COUNT | NUMBER | Number of used pieces of memory on the reserved list[1] |
| MAX_USED_SIZE | NUMBER | Size of the largest used piece of memory on the reserved list[1] |
| REQUESTS | NUMBER | Number of times that the reserved list was searched for a free piece of memory[1] |
| REQUEST_MISSES | NUMBER | Number of times the reserved list did not have a free piece of memory to satisfy the request, and started flushing objects from the LRU list[1] |
| LAST_MISS_SIZE | NUMBER | Request size of the last request miss, when the reserved list did not have a free piece of memory to satisfy the request and started flushing objects from the LRU list[1] |
| MAX_MISS_SIZE | NUMBER | Request size of the largest request miss, when the reserved list did not have a free piece of memory to satisfy the request and started flushing objects from the LRU list[1] |
| REQUEST_FAILURES | NUMBER | Number of times that no memory was found to satisfy a request. If an internal flush (used to free up memory) does not meet a memory need, then the error ORA-04031 occurs. |
| LAST_FAILURE_SIZE | NUMBER | Request size of the last failed request. If an internal flush (used to free up memory) does not meet a memory need, then the error ORA-04031 occurs.[2] |
| ABORTED_REQUEST_THRESHOLD | NUMBER | Minimum size of a request which signals an ORA-04031 error without flushing objects[2] |
| ABORTED_REQUESTS | NUMBER | Number of requests that signalled an ORA-04031 error without flushing objects[2] |
| LAST_ABORTED_SIZE | NUMBER | Last size of the request that returned an ORA-04031 error without flushing objects from the LRU list[2] |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs. |

| Column | Datatype | Description |
|---|---|---|
| | | • 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

[1] These columns are valid only if the initialization parameter SHARED_POOL_RESERVED_SIZE is set to a valid value.
[2] These columns contain values which are valid even if SHARED_POOL_RESERVED_SIZE is not set.

# V$SHARED_SERVER

V$SHARED_SERVER displays information on the shared server processes.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(4) | Name of the server |
| PADDR | RAW(4 \| 8) | Server's process address |
| STATUS | VARCHAR2(16) | Server status:<br>• EXEC - Executing SQL<br>• WAIT (ENQ) - Waiting for a lock<br>• WAIT (SEND) - Waiting to send data to user<br>• WAIT (COMMON) - Idle; waiting for a user request<br>• WAIT (RECEIVE) - Waiting for records to be shown in the client application<br>• WAIT (RESET) - Waiting for a circuit to reset after a break<br>• QUIT - Terminating |
| MESSAGES | NUMBER | Number of messages processed |
| BYTES | NUMBER | Total number of bytes in all messages |
| BREAKS | NUMBER | Number of breaks |
| CIRCUIT | RAW(4 \| 8) | Address of the circuit currently being serviced |
| IDLE | NUMBER | Total idle time (in hundredths of a second) |
| BUSY | NUMBER | Total busy time (in hundredths of a second) |
| IN_NET | NUMBER | Total incoming network wait time (in hundredths of a second) |
| OUT_NET | NUMBER | Total outgoing network wait time (in hundredths of a second) |
| REQUESTS | NUMBER | Total number of requests taken from the common queue in this server's lifetime |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SHARED_SERVER_MONITOR

V$SHARED_SERVER_MONITOR displays information for tuning the shared server.

| Column | Datatype | Description |
|---|---|---|
| MAXIMUM_CONNECTIONS | NUMBER | Highest number of virtual circuits in use at one time since the instance started. If this value reaches the value set for the CIRCUITS initialization parameter, then consider raising the value of CIRCUITS.<br>**See Also:** "CIRCUITS" |
| MAXIMUM_SESSIONS | NUMBER | Highest number of shared server sessions in use at one time since the instance started. If this reaches the value set for the SHARED_SERVER_SESSIONS initialization parameter, then consider raising the value of SHARED_SERVER_SESSIONS.<br>**See Also:** "SHARED_SERVER_SESSIONS" |
| SERVERS_STARTED | NUMBER | Total number of shared servers started since the instance started (but not including those started during startup) |
| SERVERS_TERMINATED | NUMBER | Total number of shared servers stopped by Oracle since the instance started |
| SERVERS_HIGHWATER | NUMBER | Highest number of servers running at one time since the instance started. If this value reaches the value set for the MAX_SHARED_SERVERS initialization parameter, then consider raising the value of SHARED_SERVERS.<br>**See Also:** "SHARED_SERVERS" |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SORT_SEGMENT

V$SORT_SEGMENT displays information about every sort segment in a given instance. The view is only updated when the tablespace is of the TEMPORARY type.

| Column | Datatype | Description |
|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | Name of the tablespace |
| SEGMENT_FILE | NUMBER | File number of the first extent |
| SEGMENT_BLOCK | NUMBER | Block number of the first extent |
| EXTENT_SIZE | NUMBER | Extent size |
| CURRENT_USERS | NUMBER | Number of active users of the segment |
| TOTAL_EXTENTS | NUMBER | Total number of extents in the segment |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_BLOCKS | NUMBER | Total number of blocks in the segment |
| USED_EXTENTS | NUMBER | Extents allocated to active sorts |
| USED_BLOCKS | NUMBER | Blocks allocated to active sorts |
| FREE_EXTENTS | NUMBER | Extents not allocated to any sort |
| FREE_BLOCKS | NUMBER | Blocks not allocated to any sort |
| ADDED_EXTENTS | NUMBER | Number of extent allocations |
| EXTENT_HITS | NUMBER | Number of times an unused extent was found in the pool |
| FREED_EXTENTS | NUMBER | Number of deallocated extents |
| FREE_REQUESTS | NUMBER | Number of requests to deallocate |
| MAX_SIZE | NUMBER | Maximum number of extents ever used |
| MAX_BLOCKS | NUMBER | Maximum number of blocks ever used |
| MAX_USED_SIZE | NUMBER | Maximum number of extents used by all sorts |
| MAX_USED_BLOCKS | NUMBER | Maximum number of blocks used by all sorts |
| MAX_SORT_SIZE | NUMBER | Maximum number of extents used by an individual sort |
| MAX_SORT_BLOCKS | NUMBER | Maximum number of blocks used by an individual sort |
| RELATIVE_FNO | NUMBER | Relative file number of the sort segment header |
| TS# | NUMBER | Tablespace number |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SPPARAMETER

V$SPPARAMETER displays information about the contents of the server parameter file. If a server parameter file was not used to start the instance, then each row of the view will contain FALSE in the ISSPECIFIED column.

| Column | Datatype | Description |
|---|---|---|
| FAMILY | VARCHAR2(80) | For internal use only |
| SID | VARCHAR2(80) | SID for which the parameter is defined. |
| NAME | VARCHAR2(80) | Name of the parameter |
| TYPE | VARCHAR2(11) | Parameter type |

| Column | Datatype | Description |
|---|---|---|
| VALUE | VARCHAR2(255) | Parameter value (null if a server parameter file was not used to start the instance) |
| DISPLAY_VALUE | VARCHAR2(255) | Parameter value in a user-friendly format. For example, if the VALUE column shows the value 262144 for a big integer parameter, then the DISPLAY_VALUE column will show the value 256K. |
| ISSPECIFIED | VARCHAR2(6) | Indicates whether the parameter was specified in the server parameter file (TRUE) or not (FALSE) |
| ORDINAL | NUMBER | Position (ordinal number) of the parameter value (0 if a server parameter file was not used to start the instance). Useful only for parameters whose values are lists of strings. |
| UPDATE_COMMENT | VARCHAR2(255) | Comments associated with the most recent update (null if a server parameter file was not used to start the instance) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL

V$SQL lists statistics on shared SQL areas without the GROUP BY clause and contains one row for each child of the original SQL text entered. Statistics displayed in V$SQL are normally updated at the end of query execution. However, for long running queries, they are updated every 5 seconds. This makes it easy to see the impact of long running SQL statements while they are still in progress.

| Column | Datatype | Description |
|---|---|---|
| SQL_TEXT | VARCHAR2(1000) | First thousand characters of the SQL text for the current cursor |
| SQL_FULLTEXT | CLOB | Full text for the SQL statement exposed as a CLOB column. The full text of a SQL statement can be retrieved using this column instead of joining with the V$SQLTEXT dynamic performance view. |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| SHARABLE_MEM | NUMBER | Amount of shared memory used by the child cursor (in bytes) |
| PERSISTENT_MEM | NUMBER | Fixed amount of memory used for the lifetime of the child cursor (in bytes) |
| RUNTIME_MEM | NUMBER | Fixed amount of memory required during the execution of the child cursor |
| SORTS | NUMBER | Number of sorts that were done for the child cursor |
| LOADED_VERSIONS | NUMBER | Indicates whether the context heap is loaded (1) or not (0) |

| Column | Datatype | Description |
|---|---|---|
| OPEN_VERSIONS | NUMBER | Indicates whether the child cursor is locked (1) or not (0) |
| USERS_OPENING | NUMBER | Number of users that have any of the child cursors open |
| FETCHES | NUMBER | Number of fetches associated with the SQL statement |
| EXECUTIONS | NUMBER | Number of executions that took place on this object since it was brought into the library cache |
| PX_SERVERS_EXECUTIONS | NUMBER | Total number of executions performed by parallel execution servers (0 when the statement has never been executed in parallel) |
| END_OF_FETCH_COUNT | NUMBER | Number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less or equal to the value of the EXECUTIONS column. |
| USERS_EXECUTING | NUMBER | Number of users executing the statement |
| LOADS | NUMBER | Number of times the object was either loaded or reloaded |
| FIRST_LOAD_TIME | VARCHAR2(19) | Timestamp of the parent creation time |
| INVALIDATIONS | NUMBER | Number of times this child cursor has been invalidated |
| PARSE_CALLS | NUMBER | Number of parse calls for this child cursor |
| DISK_READS | NUMBER | Number of disk reads for this child cursor |
| DIRECT_WRITES | NUMBER | Number of direct writes for this child cursor |
| BUFFER_GETS | NUMBER | Number of buffer gets for this child cursor |
| APPLICATION_WAIT_TIME | NUMBER | Application wait time (in microseconds) |
| CONCURRENCY_WAIT_TIME | NUMBER | Concurrency wait time (in microseconds) |
| CLUSTER_WAIT_TIME | NUMBER | Cluster wait time (in microseconds) |
| USER_IO_WAIT_TIME | NUMBER | User I/O Wait Time (in microseconds) |
| PLSQL_EXEC_TIME | NUMBER | PL/SQL execution time (in microseconds) |
| JAVA_EXEC_TIME | NUMBER | Java execution time (in microseconds) |
| ROWS_PROCESSED | NUMBER | Total number of rows the parsed SQL statement returns |
| COMMAND_TYPE | NUMBER | Oracle command type definition |
| OPTIMIZER_MODE | VARCHAR2(10) | Mode under which the SQL statement was executed |
| OPTIMIZER_COST | NUMBER | Cost of this query given by the optimizer |

| Column | Datatype | Description |
|---|---|---|
| OPTIMIZER_ENV | RAW(2000) | Optimizer environment |
| OPTIMIZER_ENV_HASH_VALUE | NUMBER | Hash value for the optimizer environment |
| PARSING_USER_ID | NUMBER | User ID of the user who originally built this child cursor |
| PARSING_SCHEMA_ID | NUMBER | Schema ID that was used to originally build this child cursor |
| PARSING_SCHEMA_NAME | VARCHAR2(30) | Schema name that was used to originally build this child cursor |
| KEPT_VERSIONS | NUMBER | Indicates whether this child cursor has been marked to be kept pinned in the cache using the DBMS_SHARED_POOL package |
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| TYPE_CHK_HEAP | RAW(4 \| 8) | Descriptor of the type check heap for this child cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache |
| OLD_HASH_VALUE | NUMBER | Old SQL hash value |
| PLAN_HASH_VALUE | NUMBER | Numeric representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). |
| FULL_PLAN_HASH_VALUE | NUMBER | Numeric representation of the complete SQL plan for this cursor. Comparing one FULL_PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). Note that the FULL_PLAN_HASH_VALUE cannot be compared across databases releases. It is not backward compatible. |
| CHILD_NUMBER | NUMBER | Number of this child cursor |
| SERVICE | VARCHAR2(64) | Service name |
| SERVICE_HASH | NUMBER | Hash value for the name listed in the SERVICEcolumn |
| MODULE | VARCHAR2(64) | Contains the name of the module that was executing when the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_MODULE |
| MODULE_HASH | NUMBER | Hash value of the module listed in the MODULE column |
| ACTION | VARCHAR2(64) | Contains the name of the action that was executing when the SQL statement was first parsed, which is set by calling DBMS_APPLICATION_INFO.SET_ACTION |
| ACTION_HASH | NUMBER | Hash value of the action listed in the ACTION column |
| SERIALIZABLE_ABORTS | NUMBER | Number of times the transaction failed to serialize, producing ORA-08177 errors, per cursor |
| OUTLINE_CATEGORY | VARCHAR2(64) | If an outline was applied during construction of the cursor, then this column displays the category of that outline. Otherwise the column is left blank. |

| Column | Datatype | Description |
|---|---|---|
| CPU_TIME | NUMBER | CPU time (in microseconds) used by this cursor for parsing, executing, and fetching |
| ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) used by this cursor for parsing, executing, and fetching. If the cursor uses parallel execution, then ELAPSED_TIME is the cumulative time for the query coordinator, plus all parallel query slave processes. |
| OUTLINE_SID | NUMBER | Outline session identifier |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| SQLTYPE | NUMBER | Denotes the version of the SQL language used for this statement |
| REMOTE | VARCHAR2(1) | Indicates whether the cursor is remote mapped (Y) or not (N) |
| OBJECT_STATUS | VARCHAR2(19) | Status of the cursor:<br>• VALID - Valid, authorized without errors<br>• VALID_AUTH_ERROR - Valid, authorized with authorization errors<br>• VALID_COMPILE_ERROR - Valid, authorized with compilation errors<br>• VALID_UNAUTH - Valid, unauthorized<br>• INVALID_UNAUTH - Invalid, unauthorized<br>• INVALID - Invalid, unauthorized but keep the timestamp |
| LITERAL_HASH_VALUE | NUMBER | Hash value of the literals which are replaced with system-generated bind variables and are to be matched, when CURSOR_SHARING is used. This is not the hash value for the SQL statement. If CURSOR_SHARING is not used, then the value is 0. |
| LAST_LOAD_TIME | VARCHAR2(19) | Time at which the query plan was loaded into the library cache |
| IS_OBSOLETE | VARCHAR2(1) | Indicates whether the cursor has become obsolete (Y) or not (N). This can happen if the number of child cursors is too large. |
| IS_BIND_SENSITIVE | VARCHAR2(1) | Indicates whether the cursor is bind sensitive (Y) or not (N). A query is considered bind-sensitive if the optimizer peeked at one of its bind variable values when computing predicate selectivities and where a change in a bind variable value may cause the optimizer to generate a different plan. |
| IS_BIND_AWARE | VARCHAR2(1) | Indicates whether the cursor is bind aware (Y) or not (N). A query is considered bind-aware if it has been marked to use extended cursor sharing. The query would already have been marked as bind-sensitive. |
| IS_SHAREABLE | VARCHAR2(1) | Indicates whether the cursor can be shared (Y) or not (N) |
| CHILD_LATCH | NUMBER | Child latch number that is protecting the cursor. This column is obsolete and maintained for backward compatibility. |
| SQL_PROFILE | VARCHAR2(64) | SQL profile used for this statement, if any |
| SQL_PATCH | VARCHAR2(30) | SQL patch used for this statement, if any |
| SQL_PLAN_BASELINE | VARCHAR2(30) | SQL plan baseline used for this statement, if any |
| PROGRAM_ID | NUMBER | Program identifier |

| Column | Datatype | Description |
|---|---|---|
| PROGRAM_LINE# | NUMBER | Program line number |
| EXACT_MATCHING_SIGNATURE | NUMBER | Signature calculated on the normalized SQL text. The normalization includes the removal of white space and the uppercasing of all non-literal strings. |
| FORCE_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to FORCE |
| LAST_ACTIVE_TIME | DATE | TIme at which the query plan was last active |
| BIND_DATA | RAW(2000) | Bind data |
| TYPECHECK_MEM | NUMBER | Typecheck memory |
| IO_CELL_OFFLOAD_ELIGIBLE_BYTES | NUMBER | Number of I/O bytes which can be filtered by the Exadata storage system<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_INTERCONNECT_BYTES | NUMBER | Number of I/O bytes exchanged between Oracle Database and the storage system |
| PHYSICAL_READ_REQUESTS | NUMBER | Number of physical read I/O requests issued by the monitored SQL |
| PHYSICAL_READ_BYTES | NUMBER | Number of bytes read from disks by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS | NUMBER | Number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_BYTES | NUMBER | Number of bytes written to disks by the monitored SQL |
| OPTIMIZED_PHY_READ_REQUESTS | NUMBER | Number of physical read I/O requests from Database Smart Flash Cache issued by the monitored SQL |
| LOCKED_TOTAL | NUMBER | Total number of times the child cursor has been locked |
| PINNED_TOTAL | NUMBER | Total number of times the child cursor has been pinned |
| IO_CELL_UNCOMPRESSED_BYTES | NUMBER | Number of uncompressed bytes (that is, size after decompression) that are offloaded to the Exadata cells<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_CELL_OFFLOAD_RETURNED_BYTES | NUMBER | Number of bytes that are returned by Exadata cell through the regular I/O path<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |
| IS_REOPTIMIZABLE | VARCHAR2(1) | This columns shows whether the next execution matching this child cursor will trigger a reoptimization. The values are:<br>• Y: If the next execution will trigger a reoptimization<br>• R: If the child cursor contains reoptimization information, but will not trigger reoptimization because the cursor was compiled in reporting mode<br>• N: If the child cursor has no reoptimization information |
| IS_RESOLVED_ADAPTIVE_PLAN | VARCHAR2(1) | This column shows whether all of the adaptive parts of a plan have been resolved to the final plan. Once the plan is resolved, the plan hash value and the plan displayed by DBMS_XPLAN will not change through the end of execution. The values for this column are:<br>• NULL: If the plan is not adaptive<br>• Y: If the plan is fully resolved<br>• N: If the plan is not yet fully resolved<br>**See Also:** *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XPLAN package |
| IM_SCANS | NUMBER | Number of In-Memory Column Store (IM column store) segment scans |
| IM_SCAN_BYTES_UNCOMPRESSED | NUMBER | Uncompressed size of data scanned from the IM column store |
| IM_SCAN_BYTES_INMEMORY | NUMBER | In-memory size of data scanned from the IM column store |

**See Also:**

- "V$SQLTEXT"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SHARED_POOL package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_MODULE procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_ACTION procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XPLAN package

# V$SQL_BIND_CAPTURE

V$SQL_BIND_CAPTURE displays information on bind variables used by SQL cursors. Each row in the view contains information for one bind variable defined in a cursor. This includes:

- Reference to the cursor defining the bind variable

(hash_value, address) for the parent cursor and (hash_value, child_address) for the child cursor.

- Bind metadata

  Name, position, data type, character set ID, precision, scale, and maximum length of the bind variable.

- Bind data

  One of the bind values used for the bind variable during a past execution of its associated SQL statement. Bind values are not always captured for this view. Bind values are displayed by this view only when the type of the bind variable is simple (this excludes LONG, LOB, and ADT data types) and when the bind variable is used in the WHERE or HAVING clauses of the SQL statement.

Bind capture is disabled when the STATISTICS_LEVEL initialization parameter is set to BASIC. This view can be joined with V$SQLAREA on (HASH_VALUE, ADDRESS) and with V$SQL on (HASH_VALUE, CHILD_ADDRESS).

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the parent cursor |
| HASH_VALUE | NUMBER | Hash value of the parent cursor in the library cache. The hash value is a fixed index for the view and should always be used to speed up access to the view. |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| CHILD_NUMBER | NUMBER | Child cursor number |
| NAME | VARCHAR2(30) | Name of the bind variable |
| POSITION | NUMBER | Position of the bind variable in the SQL statement |
| DUP_POSITION | NUMBER | If the binding is performed by name and the bind variable is duplicated, then this column gives the position of the primary bind variable. |
| DATATYPE | NUMBER | Internal identifier for the bind data type. Beginning in Oracle Database 12*c*, a number representing a PL/SQL data type can appear in this column. |
| DATATYPE_STRING | VARCHAR2(15) | Textual representation of the bind data type. Beginning in Oracle Database 12*c*, a text representation of a PL/SQL-only data type can appear in this column. If the actual data type is a PL/SQL sub type, the name of the data type, not the sub type will be displayed. |
| CHARACTER_SID | NUMBER | National character set identifier |
| PRECISION | NUMBER | Precision (for numeric binds) |
| SCALE | NUMBER | Scale (for numeric binds) |
| MAX_LENGTH | NUMBER | Maximum bind length |
| WAS_CAPTURED | VARCHAR2(3) | Indicates whether the bind value was captured (YES) or not (NO) |

| Column | Datatype | Description |
|---|---|---|
| LAST_CAPTURED | DATE | Date when the bind value was captured. Bind values are captured when SQL statements are executed. To limit the overhead, binds are captured at most every 15 minutes for a given cursor. |
| VALUE_STRING | VARCHAR2(4000) | Value of the bind represented as a string |
| VALUE_ANYDATA | ANYDATA | Value of the bind represented using the ANYDATA data type. This representation is useful to programmatically decode the value of the bind variable. This column is NULL if a PL/SQL-only data type appears in the DATATYPE column. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "STATISTICS_LEVEL"
- "V$SQLAREA"
- "V$SQL"

# V$SQL_BIND_DATA

V$SQL_BIND_DATA describes, for each distinct bind variable in each cursor owned by the session querying this view:

- Actual bind data, if the bind variable is user defined
- The underlying literal, if the CURSOR_SHARING parameter is set to FORCE and the bind variable is system generated. (System-generated binds have a value of 256 in the SHARED_FLAG2 column.)

| Column | Datatype | Description |
|---|---|---|
| CURSOR_NUM | NUMBER | Cursor number for this bind |
| POSITION | NUMBER | Bind position |
| DATATYPE | NUMBER | Internal identifier for the bind data type. Beginning in Oracle Database 12*c*, a number representing a PL/SQL data type can appear in this column. |
| SHARED_MAX_LEN | NUMBER | Shared maximum length for this bind from the shared cursor object associated with this bind |
| PRIVATE_MAX_LEN | NUMBER | Private maximum length for this bind sent from the client |

| Column | Datatype | Description |
|---|---|---|
| ARRAY_SIZE | NUMBER | Maximum number of array elements (for array binds only) |
| PRECISION | NUMBER | Precision (for numeric binds) |
| SCALE | NUMBER | Scale (for numeric binds) |
| SHARED_FLAG | NUMBER | Shared bind data flags |
| SHARED_FLAG2 | NUMBER | Shared bind data flags (continued) |
| BUF_ADDRESS | RAW(4 \| 8) | Bind buffer memory address |
| BUF_LENGTH | NUMBER | Bind buffer length |
| VAL_LENGTH | NUMBER | Actual bind value length |
| BUF_FLAG | NUMBER | Bind buffer flags |
| INDICATOR | NUMBER | Bind indicator |
| VALUE | VARCHAR2(4000) | Contents of the bind buffer |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"CURSOR_SHARING"

# V$SQL_BIND_METADATA

V$SQL_BIND_METADATA describes, for each distinct bind variable in each cursor owned by the session querying this view:

- Bind metadata provided by the client, if the bind variable is user defined
- Metadata based on the underlying literal, if the CURSOR_SHARING parameter is set to FORCE and the bind variable is system-generated.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Memory address of the child cursor that owns this bind variable |
| POSITION | NUMBER | Bind position |
| DATATYPE | NUMBER | Internal identifier for the bind data type. Beginning in Oracle Database 12*c*, a number representing a PL/SQL data type can appear in this column. |

| Column | Datatype | Description |
|---|---|---|
| MAX_LENGTH | NUMBER | Maximum length of the bind value |
| ARRAY_LEN | NUMBER | Maximum number of array elements (for array binds only) |
| BIND_NAME | VARCHAR2(30) | User-defined or system-generated bind variable name (if used) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"CURSOR_SHARING"

# V$SQL_CS_HISTOGRAM

V$SQL_CS_HISTOGRAM summarizes the monitoring information stored by adaptive cursor sharing. This information is used to decide whether to enable extended cursor sharing for a query. It is stored in a histogram, whose bucket's contents are exposed by this view.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| CHILD_NUMBER | NUMBER | Number of the child cursor being monitored |
| BUCKET_ID | NUMBER | Bucket number of the monitoring histogram |
| COUNT | NUMBER | Value in this bucket of the histogram |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_CS_SELECTIVITY

V$SQL_CS_SELECTIVITY exposes the valid selectivity ranges for a child cursor in extended cursor sharing mode. A valid range consists of a low and high value for each predicate containing binds. Each predicate's selectivity (with the current bind value)

must fall between the corresponding low and high values in order for the child cursor to be shared.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| CHILD_NUMBER | NUMBER | Number of the child cursor |
| PREDICATE | VARCHAR2(40) | Predicate whose selectivity must fall between low and high values |
| RANGE_ID | NUMBER | Identifier for the range used to match up the low and high values for multiple predicates |
| LOW | VARCHAR2(10) | Lower bound for allowable selectivity |
| HIGH | VARCHAR2(10) | Upper bound for allowable selectivity |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_CS_STATISTICS

V$SQL_CS_STATISTICS contains the raw execution statistics used by the monitoring component of adaptive cursor sharing. A sample of the executions is monitored. This view exposes which executions were sampled, and what the statistics were for those executions. The statistics are cumulative for each distinct set of bind values.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| CHILD_NUMBER | NUMBER | Number of the child cursor being monitored |
| BIND_SET_HASH_VALUE | NUMBER | Hash of the values of the binds |
| PEEKED | VARCHAR2(1) | Indicates if this is the bind set used to build the cursor (Y) or not (N) |
| EXECUTIONS | NUMBER | Number of times this bind set has been executed and monitored |
| ROWS_PROCESSED | NUMBER | Cumulative number of rows processed by all row sources in the plan over all monitored executions with this bind set |

| Column | Datatype | Description |
|---|---|---|
| BUFFER_GETS | NUMBER | Cumulative number of buffer gets over all monitored executions with this bind set |
| CPU_TIME | NUMBER | Cumulative CPU time (in microseconds) used by this cursor for monitored executions with this bind set |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_CURSOR

V$SQL_CURSOR displays debugging information for each cursor associated with the session querying this view.

| Column | Datatype | Description |
|---|---|---|
| CURNO | NUMBER | Cursor number |
| FLAG | NUMBER | Flags set in the cursor |
| STATUS | VARCHAR2(9) | Status of the cursor; that is, what state the cursor is in |
| PARENT_HANDLE | RAW(4 \| 8) | Pointer to the parent cursor handle |
| PARENT_LOCK | RAW(4 \| 8) | Pointer to the parent cursor lock |
| CHILD_LOCK | RAW(4 \| 8) | Pointer to the child cursor lock |
| CHILD_PIN | RAW(4 \| 8) | Pointer to the child cursor pin |
| PERS_HEAP_MEM | NUMBER | Total amount of memory allocated from persistent heap for this cursor |
| WORK_HEAP_MEM | NUMBER | Total amount of memory allocated from the work heap for this cursor |
| BIND_VARS | NUMBER | Total number of bind positions in the query currently parsed into this cursor |
| DEFINE_VARS | NUMBER | Total number of define variables in the query currently parsed into this cursor |
| BIND_MEM_LOC | VARCHAR2(64) | Which memory heap the bind variables are stored in: either the UGA or the CGA |
| INST_FLAG | VARCHAR2(64) | Instantiation object flags |
| INST_FLAG2 | VARCHAR2(64) | Instantiation object flags (continued) |
| CHILD_HANDLE | RAW(4 \| 8) | Pointer to the child cursor handle |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_JOIN_FILTER

V$SQL_JOIN_FILTER displays performance information about the characteristics of join filters when they are used for a parallel cursor. (A join filter is a bitmap filter applied to table rows before a join operation in order to avoid parallel communication.)

| Column | Datatype | Description |
|---|---|---|
| QC_SESSION_ID | NUMBER | QC (Query Coordinator) session ID of the given cursor for the given parallel query |
| QC_INSTANCE_ID | NUMBER | QC (Query Coordinator) instance ID of the given cursor for the given parallel query |
| SQL_PLAN_HASH_VALUE | NUMBER | SQL plan hash value of the given cursor for the given parallel query |
| FILTER_ID | NUMBER | An ID that identifies the join filter in the given cursor and corresponds to the filter ID in the execution plan |
| LENGTH | NUMBER | Total size of the join filter field |
| BITS_SET | NUMBER | Number of bits set in this filter |
| FILTERED | NUMBER | Number of rows seen by the join filter |
| PROBED | NUMBER | Number of rows of the right table that have been tested against the bitmap filter. This is the sum of the filtered rows plus the non-filtered rows. |
| ACTIVE | NUMBER | Whether the filter is active (Yes) or not (No) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_MONITOR

V$SQL_MONITOR displays SQL statements whose execution have been (or are being) monitored by Oracle. An entry is created in V$SQL_MONITOR every time the execution of a SQL statement is being monitored. SQL monitoring is automatically started when a SQL statement runs parallel or when it has consumed at least 5 seconds of CPU or I/O time.

When the SQL statement being monitored is executing, statistics in V$SQL_MONITOR are generally refreshed in near real time, once every second. Once the execution ends, monitoring information is not deleted immediately. Instead, it is kept in V$SQL_MONITOR for at least one minute. The entry will eventually be deleted to reclaim its space as new statements are monitored.

| Column | Datatype | Description |
|---|---|---|
| KEY | NUMBER | Artificial join key to efficiently join V$SQL_MONITOR with its corresponding plan level monitoring statistics stored in V$SQL_PLAN_MONITOR |
| REPORT_ID | NUMBER | Unique ID of the XML report stored in Automatic Workload Repository (AWR) for this monitored entity |
| STATUS | VARCHAR2(19) | SQL execution status:<br>• QUEUED - SQL statement is queued<br>• EXECUTING - SQL statement is still executing<br>• DONE (ERROR) - Execution terminated with an error<br>• DONE (FIRST N ROWS) - Execution terminated by the application before all rows were fetched<br>• DONE (ALL ROWS) - Execution terminated and all rows were fetched<br>• DONE - Execution terminated (parallel execution) |
| USER# | NUMBER | User ID of the database user who issued the SQL being monitored |
| USERNAME | VARCHAR2(30) | User name of the database user who issued the SQL being monitored |
| MODULE | VARCHAR2(64) | Name of the executing module when sampled, as set by the DBMS_APPLICATION_INFO.SET_MODULE procedure |
| ACTION | VARCHAR2(64) | Name of the executing action when sampled, as set by the DBMS_APPLICATION_INFO.SET_ACTION procedure |
| SERVICE_NAME | VARCHAR2(64) | Service name of the user session |
| CLIENT_IDENTIFIER | VARCHAR2(64) | Client identifier from the user session |
| CLIENT_INFO | VARCHAR2(64) | Client information for the user session |
| PROGRAM | VARCHAR2(48) | Name of the operating system program that issued the monitored SQL |
| PLSQL_ENTRY_OBJECT_ID | NUMBER | Object ID of the top-most PL/SQL subprogram on the stack; NULL if there is no PL/SQL subprogram on the stack |
| PLSQL_ENTRY_SUBPROGRAM_ID | NUMBER | Subprogram ID of the top-most PL/SQL subprogram on the stack; NULL if there is no PL/SQL subprogram on the stack |
| PLSQL_OBJECT_ID | NUMBER | Object ID of the currently executing PL/SQL subprogram; NULL if executing SQL |
| PLSQL_SUBPROGRAM_ID | NUMBER | Subprogram ID of the currently executing PL/SQL object; NULL if executing SQL |
| FIRST_REFRESH_TIME | DATE | Time when monitoring of the SQL statement started, generally a few seconds after execution start time |

| Column | Datatype | Description |
|---|---|---|
| LAST_REFRESH_TIME | DATE | Time when statistics in V$SQL_MONITOR were last updated for the SQL statement. Statistics are generally refreshed every second when the statement executes. |
| REFRESH_COUNT | NUMBER | Number of times V$SQL_MONITOR statistics have been refreshed (generally once every second when the SQL statement executes) |
| DBOP_EXEC_ID | NUMBER | Database operation execution identifier for the current execution. If the type is SQL, the DBOP_EXEC_ID will be NULL. |
| DBOP_NAME | VARCHAR2(30) | Database operation name. If the type is SQL, the DBOP_NAME will be NULL. |
| SID | NUMBER | Session identifier executing (or having executed) the SQL statement being monitored |
| PROCESS_NAME | VARCHAR2(5) | Process name identifier executing (or having executed)the statement; ora if the process is foreground, else the background process name (for example, p001 for PX server p001) |
| SQL_ID | VARCHAR2(13) | SQL identifier of the statement being monitored |
| SQL_TEXT | VARCHAR2(2000) | Up to the first 2000 characters of the text of the SQL being monitored |
| IS_FULL_SQLTEXT | VARCHAR2(1) | Indicates whether the SQL_TEXT column has the entire SQL text (Y) or not (N) |
| SQL_EXEC_START | DATE | Time when the execution started |
| SQL_EXEC_ID | NUMBER | Execution identifier. Together, the three columns SQL_ID, SQL_EXEC_START, and SQL_EXEC_ID represent the execution key. The execution key is used to uniquely identify one execution of the SQL statement. |
| SQL_PLAN_HASH_VALUE | NUMBER | SQL plan hash value |
| SQL_FULL_PLAN_HASH_VALUE | NUMBER | Numeric representation of the complete SQL plan for this cursor. Comparing one SQL_FULL_PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). Note that the SQL_FULL_PLAN_HASH_VALUE cannot be compared across databases releases. It is not backward compatible. |
| EXACT_MATCHING_SIGNATURE | NUMBER | Signature calculated on the normalized SQL text. The normalization includes the removal of white space and the uppercasing of all non-literal strings. |
| FORCE_MATCHING_SIGNATURE | NUMBER | Same as EXACT_MATCHING_SIGNATURE but literals in the SQL text are replaced by binds |
| SQL_CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor (can be used with SQL_ID to join with V$SQL) |
| SESSION_SERIAL# | NUMBER | Session serial number executing the statement being monitored |
| PX_IS_CROSS_INSTANCE | VARCHAR2(1) | Indicates whether the SQL statement ran parallel across multiple instances (Y) or not (N) |

| Column | Datatype | Description |
|---|---|---|
| PX_MAXDOP | NUMBER | Maximum degree of parallelism for any plan operation executed on behalf of the monitored SQL |
| PX_MAXDOP_INSTANCES | NUMBER | Number of database instances touched at the maximum degree of parallelism |
| PX_SERVERS_REQUESTED | NUMBER | Total number of parallel execution servers requested to execute the monitored SQL |
| PX_SERVERS_ALLOCATED | NUMBER | Actual number of parallel execution servers allocated to execute the query |
| PX_SERVER# | NUMBER | Logical parallel execution server process number executing (or having executed) the statement being monitored; NULL if this monitoring entry is not associated with an execution server. This is a logical number within the parallel server set (see SERVER# in V$PX_SESSION). |
| PX_SERVER_GROUP | NUMBER | Logical parallel execution server group number to which PX_SERVER# belongs (see SERVER_GROUP in V$PX_SESSION); NULL if this monitoring entry is not associated with a parallel execution server. This value is generally 1 unless the SQL statement has one or more parallel sub-queries. |
| PX_SERVER_SET | NUMBER | Number (1 or 2) of the logical set of parallel execution servers to which PX_SERVER# belongs (see SERVER_SET in V$PX_SESSION); NULL if this monitoring entry is not associated with a parallel execution server |
| PX_QCINST_ID | NUMBER | Instance identifier where the parallel execution coordinator runs; NULL if PX_SERVER# is NULL |
| PX_QCSID | NUMBER | Session identifier for the parallel execution coordinator; NULL if PX_SERVER# is NULL |
| ERROR_NUMBER | VARCHAR2(40) | Error number encountered in case a SQL fails to execute successfully (for example, 932 in case of ORA-00932) |
| ERROR_FACILITY | VARCHAR2(4) | Error facility in case a SQL fails to execute successfully (for example, ORA in case of ORA-00932) |
| ERROR_MESSAGE | VARCHAR2(256) | Detailed error message displayed corresponding to the error number and error facility when a SQL fails to execute successfully |
| BINDS_XML | CLOB | Information about bind variables used with the SQL, such as name, position, value, data type, and so on (stored in XML format) |
| OTHER_XML | CLOB | Additional information about SQL execution stored in XML format |
| ELAPSED_TIME | NUMBER | Elapsed time (in microseconds); updated as the statement executes |
| QUEUING_TIME | NUMBER | Duration of time (in microseconds) spent by SQL in the statement queue |
| CPU_TIME | NUMBER | CPU time (in microseconds); updated as the statement executes |

| Column | Datatype | Description |
|---|---|---|
| FETCHES | NUMBER | Number of fetches associated with the SQL statement; updated as the statement executes |
| BUFFER_GETS | NUMBER | Number of buffer get operations; updated as the statement executes |
| DISK_READS | NUMBER | Number of disk reads; updated as the statement executes |
| DIRECT_WRITES | NUMBER | Number of direct writes; updated as the statement executes |
| IO_INTERCONNECT_BYTES | NUMBER | Number of I/O bytes exchanged between Oracle Database and the storage system |
| PHYSICAL_READ_REQUESTS | NUMBER | Number of physical read I/O requests issued by the monitored SQL |
| PHYSICAL_READ_BYTES | NUMBER | Number of bytes read from disks by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS | NUMBER | Number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_BYTES | NUMBER | Number of bytes written to disks by the monitored SQL |
| APPLICATION_WAIT_TIME | NUMBER | Application wait time (in microseconds); updated as the statement executes |
| CONCURRENCY_WAIT_TIME | NUMBER | Concurrency wait time (in microseconds); updated as the statement executes |
| CLUSTER_WAIT_TIME | NUMBER | Cluster wait time (in microseconds); updated as the statement executes |
| USER_IO_WAIT_TIME | NUMBER | User I/O Wait Time (in microseconds); updated as the statement executes |
| PLSQL_EXEC_TIME | NUMBER | PL/SQL execution time (in microseconds); updated as the statement executes |
| JAVA_EXEC_TIME | NUMBER | Java execution time (in microseconds); updated as the statement executes |
| RM_LAST_ACTION | VARCHAR2(48) | The most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following:<br>• CANCEL_SQL<br>• KILL_SESSION<br>• LOG_ONLY<br>• SWITCH TO <CG NAME><br>For the last value, <CG NAME> is the name of the consumer group that the SQL operation was switched to. If the Resource Plan has since been changed then <CG NAME> is the ID of the consumer group. |
| RM_LAST_ACTION_REASON | VARCHAR2(30) | The reason for the most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following:<br>• SWITCH_CPU_TIME<br>• SWITCH_IO_REQS<br>• SWITCH_IO_MBS |

| Column | Datatype | Description |
|---|---|---|
| | | • SWITCH_ELAPSED_TIME<br>• SWITCH_IO_LOGICAL |
| RM_LAST_ACTION_TIME | DATE | The time of the most recent action that was taken on this SQL operation by Resource Manager |
| RM_CONSUMER_GROUP | VARCHAR2(30) | The current consumer group for this SQL operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CON_NAME | VARCHAR2(30) | Container name of the object. The value of this column is NULL in non-CDBs. |
| ECID | VARCHAR2(64) | Execution context identifier (sent by Application Server) |
| IS_ADAPTIVE_PLAN | VARCHAR2(1) | Indicates whether the statistics are from an adaptive plan (Y) or not (N). |
| IS_FINAL_PLAN | VARCHAR2(1) | Indicates whether the statistics are from the final plan (Y) or not (N). |

**See Also:**

- "V$SQL_PLAN_MONITOR"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_MODULE procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_ACTION procedure

# V$SQL_MONITOR_SESSTAT

V$SQL_MONITOR_SESSTAT displays information about statistics that are exposed in V$SESSTAT.

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

| Column | Datatype | Description |
|---|---|---|
| KEY | NUMBER | Artificial join key to efficiently join with other database operation views |
| STATISTIC# | NUMBER | Statistic number |
| VALUE | NUMBER | Value associated with the statistic |

**See Also:**

"V$SESSTAT"

# V$SQL_MONITOR_STATNAME

V$SQL_MONITOR_STATNAME provides information about the plan line statistics exposed in V$SQL_PLAN_MONITOR. A plan line statistic is identified by its group ID (column GROUP_ID) and its ID (column ID).

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| ID | NUMBER | Plan line statistic ID |
| GROUP_ID | NUMBER | Plan line statistic group ID |
| NAME | VARCHAR2(40) | Short name for the statistic |
| DESCRIPTION | VARCHAR2(200) | Short description for the statistic |
| TYPE | NUMBER | Reserved for internal use |
| FLAGS | NUMBER | Reserved for internal use |

**See Also:**

"V$SQL_PLAN_MONITOR"

# V$SQL_OPTIMIZER_ENV

V$SQL_OPTIMIZER_ENV displays the contents of the optimizer environment used to build the execution plan of a SQL cursor. This view can be joined with V$SQLAREA on (HASH_VALUE, ADDRESS) and with V$SQL on (HASH_VALUE, CHILD_ADDRESS).

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the parent cursor |

| Column | Datatype | Description |
|---|---|---|
| HASH_VALUE | NUMBER | Hash value of the parent cursor in the library cache. The hash value is the fixed index for the view and should be used when querying V$SQL_OPTIMIZER_ENV to avoid scanning the entire library cache. |
| SQL_ID | VARCHAR2(13) | SQL identifier |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| CHILD_NUMBER | NUMBER | Child cursor number |
| ID | NUMBER | Unique identifier of the parameter in the optimizer environment |
| NAME | VARCHAR2(40) | Name of the parameter |
| ISDEFAULT | VARCHAR2(3) | Indicates whether the parameter is set to the default value (YES) or not (NO) |
| VALUE | VARCHAR2(25) | Value of the parameter |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_PLAN

V$SQL_PLAN contains the execution plan information for each child cursor loaded in the library cache.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache. The two columns ADDRESS and HASH_VALUE can be used to join with V$SQLAREA to add the cursor-specific information. |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| PLAN_HASH_VALUE | NUMBER | Numerical representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). |
| FULL_PLAN_HASH_VALUE | NUMBER | Numeric representation of the complete SQL plan for this cursor. Comparing one FULL_PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). Note that the FULL_PLAN_HASH_VALUE cannot be compared across databases releases. It is not backward compatible. |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |

| Column | Datatype | Description |
|---|---|---|
| CHILD_NUMBER | NUMBER | Number of the child cursor that uses this execution plan. The columns ADDRESS, HASH_VALUE, and CHILD_NUMBER can be used to join with V$SQL to add the child cursor-specific information. |
| TIMESTAMP | DATE | Date and time when the execution plan was generated |
| OPERATION | VARCHAR2(30) | Name of the internal operation performed in this step (for example, TABLE ACCESS) |
| OPTIONS | VARCHAR2(30) | A variation on the operation described in the OPERATION column (for example, FULL) |
| OBJECT_NODE | VARCHAR2(40) | Name of the database link used to reference the object (a table name or view name). For local queries that use parallel execution, this column describes the order in which output from operations is consumed. |
| OBJECT# | NUMBER | Object number of the table or the index |
| OBJECT_OWNER | VARCHAR2(30) | Name of the user who owns the schema containing the table or index |
| OBJECT_NAME | VARCHAR2(30) | Name of the table or index |
| OBJECT_ALIAS | VARCHAR2(65) | Alias for the object |
| OBJECT_TYPE | VARCHAR2(20) | Type of the object |
| OPTIMIZER | VARCHAR2(20) | Current mode of the optimizer for the first row in the plan (statement line), for example, CHOOSE. When the operation is a database access (for example, TABLE ACCESS), this column indicates whether or not the object is analyzed. |
| ID | NUMBER | A number assigned to each step in the execution plan |
| PARENT_ID | NUMBER | ID of the next execution step that operates on the output of the current step |
| DEPTH | NUMBER | Depth (or level) of the operation in the tree. It is not necessary to issue a CONNECT BY statement to get the level information, which is generally used to indent the rows from the PLAN_TABLE table. The root operation (statement) is level 0. |
| POSITION | NUMBER | Order of processing for all operations that have the same PARENT_ID |
| SEARCH_COLUMNS | NUMBER | Number of index columns with start and stop keys (that is, the number of columns with matching predicates) |
| COST | NUMBER | Cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| CARDINALITY | NUMBER | Estimate, by the cost-based optimizer, of the number of rows produced by the operation |
| BYTES | NUMBER | Estimate, by the cost-based optimizer, of the number of bytes produced by the operation |

| Column | Datatype | Description |
|---|---|---|
| OTHER_TAG | VARCHAR2(35) | Describes the contents of the OTHER column. See EXPLAIN PLAN for values. |
| PARTITION_START | VARCHAR2(64) | Start partition of a range of accessed partitions |
| PARTITION_STOP | VARCHAR2(64) | Stop partition of a range of accessed partitions |
| PARTITION_ID | NUMBER | Step that computes the pair of values of the PARTITION_START and PARTITION_STOP columns |
| OTHER | VARCHAR2(4000) | Other information specific to the execution step that users may find useful. See EXPLAIN PLAN for values. |
| DISTRIBUTION | VARCHAR2(20) | Stores the method used to distribute rows from producer query servers to consumer query servers |
| CPU_COST | NUMBER | CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| IO_COST | NUMBER | I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| TEMP_SPACE | NUMBER | Temporary space usage of the operation (sort or hash-join) as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| ACCESS_PREDICATES | VARCHAR2(4000) | Predicates used to locate rows in an access structure. For example, start or stop predicates for an index range scan. |
| FILTER_PREDICATES | VARCHAR2(4000) | Predicates used to filter rows before producing them |
| PROJECTION | VARCHAR2(4000) | Expressions produced by the operation |
| TIME | NUMBER | Elapsed time (in seconds) of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| QBLOCK_NAME | VARCHAR2(30) | Name of the query block |
| REMARKS | VARCHAR2(4000) | Remarks |
| OTHER_XML | CLOB | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML since multiple pieces of information can be stored there. This includes:<br>• Name of the schema against which the query was parsed<br>• Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan<br>• Indication of whether or not dynamic statistics were used to produce the plan |

| Column | Datatype | Description |
|---|---|---|
| | | • The outline data, a set of optimizer hints that can be used to regenerate the same plan<br>• Additional data that describes the relationship between rows in the plan table and subplans of adaptive plans. Note that in Oracle Database 12*c*, there will be extra rows in the plan table and V$SQL_PLAN. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_PLAN_MONITOR

V$SQL_PLAN_MONITOR displays plan level monitoring statistics for each SQL statement found in V$SQL_MONITOR. Each row in V$SQL_PLAN_MONITOR corresponds to an operation of the execution plan being monitored. As with V$SQL_MONITOR, statistics exposed in V$SQL_PLAN_MONITOR are generally updated every second when the statement executes. These statistics are recycled on the same basis as V$SQL_MONITOR.

To eliminate the overhead of SQL plan monitoring, statistics collected for each operation of the plan do not record timing information such as elapsed time, CPU time, or I/O time. Instead, this timing information can be estimated quite accurately by joining V$SQL_PLAN_MONITOR with V$ACTIVE_SESSION_HISTORY on SQL_ID, SQL_EXEC_START, SQL_EXEC_ID, and SQL_PLAN_LINE_ID (simply named PLAN_LINE_ID in V$SQL_PLAN_MONITOR). The result of that join is a sample of the activity performed by each operation in the plan, from which an estimate of CPU time and wait time can be derived. This can be achieved by breaking statement level monitoring time statistics found in V$SQL_MONITOR in proportion to the number of samples found in V$ACTIVE_SESSION_HISTORY for the corresponding activity type.

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| KEY | NUMBER | Foreign key to efficiently join V$SQL_PLAN_MONITOR with V$SQL_MONITOR (see V$SQL_MONITOR) |
| STATUS | VARCHAR2(19) | SQL execution status:<br>• EXECUTING - SQL statement is still executing<br>• DONE (ERROR) - Execution terminated with an error<br>• DONE (FIRST N ROWS) - Execution terminated by the application before all rows were fetched |

| Column | Datatype | Description |
|---|---|---|
| | | • DONE (ALL ROWS) - Execution terminated and all rows were fetched<br>• DONE - Execution terminated (parallel execution) |
| FIRST_REFRESH_TIME | DATE | Time when monitoring of the SQL statement started |
| LAST_REFRESH_TIME | DATE | Time when statistics were last updated for the SQL statement |
| FIRST_CHANGE_TIME | DATE | First time a row was produced by this operation |
| LAST_CHANGE_TIME | DATE | Last time a row was produced by this operation |
| REFRESH_COUNT | NUMBER | Number of times statistics have been refreshed |
| SID | NUMBER | Session identifier executing (or having executed) the SQL statement being monitored |
| PROCESS_NAME | VARCHAR2(5) | Process name identifier |
| SQL_ID | VARCHAR2(13) | SQL identifier |
| SQL_EXEC_START | DATE | Time when the execution started |
| SQL_EXEC_ID | NUMBER | Execution identifier |
| SQL_PLAN_HASH_VALUE | NUMBER | SQL plan hash value |
| SQL_CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| PLAN_PARENT_ID | NUMBER | ID of the next execution step that operates on the output of the current step |
| PLAN_LINE_ID | NUMBER | Plan line number for the entry |
| PLAN_OPERATION | VARCHAR2(30) | Plan operation name (from V$SQL_PLAN) |
| PLAN_OPTIONS | VARCHAR2(30) | Plan option name (from V$SQL_PLAN) |
| PLAN_OBJECT_OWNER | VARCHAR2(30) | Name of the user who owns the schema containing the table or index |
| PLAN_OBJECT_NAME | VARCHAR2(30) | Name of the table or index |
| PLAN_OBJECT_TYPE | VARCHAR2(20) | Type of the object |
| PLAN_DEPTH | NUMBER | Depth (or level) of the operation in the tree. It is not necessary to issue a CONNECT BY statement to get the level information, which is generally used to indent the rows from the PLAN_TABLE table. The root operation (statement) is level 0. |
| PLAN_POSITION | NUMBER | Order of processing for all operations that have the same PARENT_ID |
| PLAN_COST | NUMBER | Cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| PLAN_CARDINALITY | NUMBER | Estimate, by the cost-based optimizer, of the number of rows produced by the operation |

| Column | Datatype | Description |
|---|---|---|
| PLAN_BYTES | NUMBER | Estimate, by the cost-based optimizer, of the number of bytes produced by the operation |
| PLAN_TIME | NUMBER | Elapsed time (in seconds) of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| PLAN_PARTITION_START | VARCHAR2(64) | Start partition of a range of accessed partitions |
| PLAN_PARTITION_STOP | VARCHAR2(64) | Stop partition of a range of accessed partitions |
| PLAN_CPU_COST | NUMBER | CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| PLAN_IO_COST | NUMBER | I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| PLAN_TEMP_SPACE | NUMBER | Temporary space usage of the operation (sort or hash-join) as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is NULL. |
| STARTS | NUMBER | Number of times this operation was executed. For example, an operation is executed multiple times when it is on the right side of a nested-loop join (once for each row of the left input of that nested-loop join). |
| OUTPUT_ROWS | NUMBER | Number of rows produced by this operation since the execution started. This number is cumulated for all executions of this operation. Divide by the value of the STARTS column to compute the average number of rows per execution of the operation. Note that the value in the STARTS column is equal to or higher than the value in the OUTPUT_ROWS column. The value will usually be equal, but depending on internal optimizations a higher value may be seen. |
| IO_INTERCONNECT_BYTES | NUMBER | Number of I/O bytes exchanged between Oracle Database and the storage system. Maintained only after Oracle starts to monitor the execution. |
| PHYSICAL_READ_REQUESTS | NUMBER | Number of physical read I/O requests issued by the monitored SQL. Maintained only after Oracle starts to monitor the execution. |
| PHYSICAL_READ_BYTES | NUMBER | Number of bytes read from disks by the monitored SQL. Maintained only after Oracle starts to monitor the execution. |
| PHYSICAL_WRITE_REQUESTS | NUMBER | Number of physical write I/O requests issued by the monitored SQL. Maintained only after Oracle starts to monitor the execution. |
| PHYSICAL_WRITE_BYTES | NUMBER | Number of bytes written to disks by the monitored SQL. Maintained only after Oracle starts to monitor the execution. |
| WORKAREA_MEM | NUMBER | Amount of memory (in bytes) used by the operation when the query is executing; NULL if the execution is done. This applies |

| Column | Datatype | Description |
|---|---|---|
| | | only to operations using a work area, such as sort, hash-join, group-by, and so on. |
| WORKAREA_MAX_MEM | NUMBER | Maximum value (in bytes) for WORKAREA_MEM; NULL if the operation is not using a work area. When the execution is finished, this value will hold the maximum amount of memory consumed by this operation during the execution of the statement. |
| WORKAREA_TEMPSEG | NUMBER | Amount of temporary space (in bytes) used by the operation when the query is executing; NULL if the operation has not spilled to disk or if the execution is finished |
| WORKAREA_MAX_TEMPSEG | NUMBER | Maximum value (in bytes) for WORKAREA_TEMPSEG; NULL if this operation never spilled to disk. When the execution is done, this value will hold the maximum amount of temporary space consumed by this operation during the entire execution. |
| OTHERSTAT_GROUP_ID | NUMBER | Plan line statistic group identifier (see GROUP_ID column in V$SQL_MONITOR_STATNAME) |
| OTHERSTAT_1_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 1 of that plan line |
| OTHERSTAT_1_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_1_VALUE | NUMBER | Value of statistic number 1 of that plan line |
| OTHERSTAT_2_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 2 of that plan line |
| OTHERSTAT_2_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_2_VALUE | NUMBER | Value of statistic number 2 of that plan line |
| OTHERSTAT_3_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 3 of that plan line |
| OTHERSTAT_3_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_3_VALUE | NUMBER | Value of statistic number 3 of that plan line |
| OTHERSTAT_4_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 4 of that plan line |
| OTHERSTAT_4_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_4_VALUE | NUMBER | Value of statistic number 4 of that plan line |
| OTHERSTAT_5_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 5 of that plan line |
| OTHERSTAT_5_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_5_VALUE | NUMBER | Value of statistic number 5 of that plan line |

| Column | Datatype | Description |
|---|---|---|
| OTHERSTAT_6_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 6 of that plan line |
| OTHERSTAT_6_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_6_VALUE | NUMBER | Value of statistic number 6 of that plan line |
| OTHERSTAT_7_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 7 of that plan line |
| OTHERSTAT_7_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_7_VALUE | NUMBER | Value of statistic number 7 of that plan line |
| OTHERSTAT_8_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 8 of that plan line |
| OTHERSTAT_8_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_8_VALUE | NUMBER | Value of statistic number 8 of that plan line |
| OTHERSTAT_9_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 9 of that plan line |
| OTHERSTAT_9_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_9_VALUE | NUMBER | Value of statistic number 9 of that plan line |
| OTHERSTAT_10_ID | NUMBER | Statistic identifier (see ID column in V$SQL_MONITOR_STATNAME) for statistic number 10 of that plan line |
| OTHERSTAT_10_TYPE | NUMBER | Reserved for internal use |
| OTHERSTAT_10_VALUE | NUMBER | Value of statistic number 10 of that plan line |
| OTHER_XML | CLOB | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML because it allows multiple pieces of information to be stored, including the following:<br>• Name of the schema against which the query was parsed<br>• Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan<br>• Indication of whether or not dynamic statistics were used to produce the plan<br>• The outline data, a set of optimizer hints that can be used to regenerate the same plan |
| PLAN_OPERATION_INACTIVE | NUMBER | Indicates whether this plan operation was part of the final resolved plan |

**See Also:**

"V$SQL_MONITOR_STATNAME"

# V$SQL_PLAN_STATISTICS

V$SQL_PLAN_STATISTICS provides execution statistics at the row source level for each child cursor.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache. The two columns ADDRESS and HASH_VALUE can be used to join with V$SQLAREA to locate the parent cursor. |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent statement in the library cache |
| PLAN_HASH_VALUE | NUMBER | Numerical representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line) |
| FULL_PLAN_HASH_VALUE | NUMBER | Numeric representation of the complete SQL plan for this cursor. Comparing one FULL_PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). Note that the FULL_PLAN_HASH_VALUE cannot be compared across databases releases. It is not backward compatible. |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| CHILD_NUMBER | NUMBER | Number of the child cursor that uses this work area. The columns ADDRESS, HASH_VALUE, and CHILD_NUMBER can be used to join with V$SQL to locate the child cursor using this area. |
| OPERATION_ID | NUMBER | A number assigned to each step in the execution plan |
| EXECUTIONS | NUMBER | Number of times this cursor has been executed |
| LAST_STARTS | NUMBER | Number of times this operation has been started, during the last execution |
| STARTS | NUMBER | Number of times this operation has been started, accumulated over the past executions |
| LAST_OUTPUT_ROWS | NUMBER | Number of rows produced by the row source, during the last execution |
| OUTPUT_ROWS | NUMBER | Number of rows produced by the row source, accumulated over the past executions |
| LAST_CR_BUFFER_GETS | NUMBER | Number of buffers retrieved in consistent mode, during the last execution. Buffers are usually retrieved in consistent mode for queries. |
| CR_BUFFER_GETS | NUMBER | Number of buffers retrieved in consistent mode, accumulated over the past executions. Buffers are usually retrieved in consistent mode for queries. |

| Column | Datatype | Description |
|---|---|---|
| LAST_CU_BUFFER_GETS | NUMBER | Number of buffers retrieved in current mode, during the last execution. Buffers are retrieved in current mode for statements such as INSERT, UPDATE, and DELETE. |
| CU_BUFFER_GETS | NUMBER | Number of buffers retrieved in current mode, accumulated over the past executions. Buffers are retrieved in current mode for statements such as INSERT, UPDATE, and DELETE. |
| LAST_DISK_READS | NUMBER | Number of physical disk reads performed by the operation, during the last execution |
| DISK_READS | NUMBER | Number of physical disk reads performed by the operation, accumulated over the past executions |
| LAST_DISK_WRITES | NUMBER | Number of physical disk writes performed by the operation, during the last execution |
| DISK_WRITES | NUMBER | Number of physical disk writes performed by the operation, accumulated over the past executions |
| LAST_ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) corresponding to this operation, during the last execution |
| ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) corresponding to this operation, accumulated over the past executions |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SQLAREA"

# V$SQL_PLAN_STATISTICS_ALL

V$SQL_PLAN_STATISTICS_ALL contains memory usage statistics for row sources that use SQL memory (sort or hash-join). This view concatenates information in V$SQL_PLAN with execution statistics from V$SQL_PLAN_STATISTICS and V$SQL_WORKAREA.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache. The two columns ADDRESS and HASH_VALUE can be used to join with V$SQLAREA to add the cursor-specific information. |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent statement in the library cache |

| Column | Datatype | Description |
|---|---|---|
| PLAN_HASH_VALUE | NUMBER | Numerical representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line) |
| FULL_PLAN_HASH_VALUE | NUMBER | Numeric representation of the complete SQL plan for this cursor. Comparing one FULL_PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). Note that the FULL_PLAN_HASH_VALUE cannot be compared across databases releases. It is not backward compatible. |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| CHILD_NUMBER | NUMBER | Number of the child cursor that uses this execution plan. The columns ADDRESS, HASH_VALUE, and CHILD_NUMBER can be used to join with V$SQL to add the child cursor-specific information. |
| TIMESTAMP | DATE | Date and time when the execution plan was generated |
| OPERATION | VARCHAR2(30) | Name of the internal operation performed in this step (for example, TABLE ACCESS) |
| OPTIONS | VARCHAR2(30) | A variation on the operation described in the OPERATION column (for example, FULL) |
| OBJECT_NODE | VARCHAR2(40) | Name of the database link used to reference the object (a table name or view name). For local queries that use parallel execution, this column describes the order in which output from operations is consumed. |
| OBJECT# | NUMBER | Object number of the table or the index |
| OBJECT_OWNER | VARCHAR2(30) | Name of the user who owns the schema containing the table or index |
| OBJECT_NAME | VARCHAR2(30) | Name of the table or index |
| OBJECT_ALIAS | VARCHAR2(65) | Alias for the object |
| OBJECT_TYPE | VARCHAR2(20) | Type of the object |
| OPTIMIZER | VARCHAR2(20) | Current mode of the optimizer for the first row in the plan (statement line), for example, CHOOSE. When the operation is a database access (for example, TABLE ACCESS), this column indicates whether or not the object is analyzed. |
| ID | NUMBER | A number assigned to each step in the execution plan |
| PARENT_ID | NUMBER | ID of the next execution step that operates on the output of the current step |
| DEPTH | NUMBER | Depth (or level) of the operation in the tree. It is not necessary to issue a CONNECT BY statement to get the level information, which is generally used to indent the rows from the PLAN_TABLE table. The root operation (statement) is level 0. |
| POSITION | NUMBER | Order of processing for all operations that have the same PARENT_ID |

| Column | Datatype | Description |
|---|---|---|
| SEARCH_COLUMNS | NUMBER | Number of index columns with start and stop keys (that is, the number of columns with matching predicates) |
| COST | NUMBER | Cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| CARDINALITY | NUMBER | Estimate, by the cost-based optimizer, of the number of rows produced by the operation |
| BYTES | NUMBER | Estimate, by the cost-based optimizer, of the number of bytes produced by the operation |
| OTHER_TAG | VARCHAR2(35) | Describes the contents of the OTHER column. See EXPLAIN PLAN for values. |
| PARTITION_START | VARCHAR2(64) | Start partition of a range of accessed partitions |
| PARTITION_STOP | VARCHAR2(64) | Stop partition of a range of accessed partitions |
| PARTITION_ID | NUMBER | Step that computes the pair of values of the PARTITION_START and PARTITION_STOP columns |
| OTHER | VARCHAR2(4000) | Other information specific to the execution step that users may find useful. See EXPLAIN PLAN for values. |
| DISTRIBUTION | VARCHAR2(20) | Stores the method used to distribute rows from producer query servers to consumer query servers |
| CPU_COST | NUMBER | CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| IO_COST | NUMBER | I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| TEMP_SPACE | NUMBER | Temporary space usage of the operation (sort or hash-join) as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| ACCESS_PREDICATES | VARCHAR2(4000) | Predicates used to locate rows in an access structure. For example, start or stop predicates for an index range scan. |
| FILTER_PREDICATES | VARCHAR2(4000) | Predicates used to filter rows before producing them |
| PROJECTION | VARCHAR2(4000) | Expressions produced by the operation |
| TIME | NUMBER | Elapsed time (in seconds) of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. |
| QBLOCK_NAME | VARCHAR2(30) | Name of the query block |
| REMARKS | VARCHAR2(4000) | Remarks |

| Column | Datatype | Description |
|---|---|---|
| OTHER_XML | CLOB | Provides extra information specific to an execution step of the execution plan. The content of this column is structured using XML since multiple pieces of information can be stored there. This includes:<br>• Name of the schema against which the query was parsed<br>• Release number of the Oracle Database that produced the explain plan<br>• Hash value associated with the execution plan<br>• Name (if any) of the outline or the SQL profile used to build the execution plan<br>• Indication of whether or not dynamic statistics were used to produce the plan<br>• The outline data, a set of optimizer hints that can be used to regenerate the same plan<br>• Additional data that describes the relationship between rows in the plan table and subplans of adaptive plans. Note that in Oracle Database 12*c*, there will be extra rows in the plan table and V$SQL_PLAN. |
| EXECUTIONS | NUMBER | Number of times this cursor has been executed |
| LAST_STARTS | NUMBER | Number of times this operation has been started, during the last execution |
| STARTS | NUMBER | Number of times this operation has been started, accumulated over the past executions |
| LAST_OUTPUT_ROWS | NUMBER | Number of rows produced by the row source, during the last execution |
| OUTPUT_ROWS | NUMBER | Number of rows produced by the row source, accumulated over the past executions |
| LAST_CR_BUFFER_GETS | NUMBER | Number of buffers retrieved in consistent mode, during the last execution. Buffers are usually retrieved in consistent mode for queries. |
| CR_BUFFER_GETS | NUMBER | Number of buffers retrieved in consistent mode, accumulated over the past executions. Buffers are usually retrieved in consistent mode for queries. |
| LAST_CU_BUFFER_GETS | NUMBER | Number of buffers retrieved in current mode, during the last execution. Buffers are retrieved in current mode for statements such as INSERT, UPDATE, and DELETE. |
| CU_BUFFER_GETS | NUMBER | Number of buffers retrieved in current mode, accumulated over the past executions. Buffers are retrieved in current mode for statements such as INSERT, UPDATE, and DELETE. |
| LAST_DISK_READS | NUMBER | Number of physical disk reads performed by the operation, during the last execution |
| DISK_READS | NUMBER | Number of physical disk reads performed by the operation, accumulated over the past executions |
| LAST_DISK_WRITES | NUMBER | Number of physical disk writes performed by the operation, during the last execution |

| Column | Datatype | Description |
|---|---|---|
| DISK_WRITES | NUMBER | Number of physical disk writes performed by the operation, accumulated over the past executions |
| LAST_ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) corresponding to this operation, during the last execution |
| ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) corresponding to this operation, accumulated over the past executions |
| POLICY | VARCHAR2(10) | Sizing policy for this work area:<br>• MANUAL<br>• AUTO |
| ESTIMATED_OPTIMAL_SIZE | NUMBER | Estimated size (in KB) required by this work area to execute the operation completely in memory (optimal execution). This is either derived from optimizer statistics or from previous executions. |
| ESTIMATED_ONEPASS_SIZE | NUMBER | Estimated size (in KB) required by this work area to execute the operation in a single pass. This is either derived from optimizer statistics or from previous executions. |
| LAST_MEMORY_USED | NUMBER | Memory size (in KB) used by this work area during the last execution of the cursor |
| LAST_EXECUTION | VARCHAR2(10) | Indicates whether this work area ran using OPTIMAL, ONE PASS, or under ONE PASS memory requirement (MULTI-PASS), during the last execution of the cursor |
| LAST_DEGREE | NUMBER | Degree of parallelism used, during the last execution of the cursor |
| TOTAL_EXECUTIONS | NUMBER | Number of times this work area was active |
| OPTIMAL_EXECUTIONS | NUMBER | Number of times this work area ran in optimal mode |
| ONEPASS_EXECUTIONS | NUMBER | Number of times this work area ran in one pass mode |
| MULTIPASSES_EXECUTIONS | NUMBER | Number of times this work area ran below the one pass memory requirement |
| ACTIVE_TIME | NUMBER | Average time this work area is active (in hundredths of a second) |
| MAX_TEMPSEG_SIZE | NUMBER | Maximum temporary segment size (in bytes) created by an instantiation of this work area. This column is null if this work area has never spilled to disk. |
| LAST_TEMPSEG_SIZE | NUMBER | Temporary segment size (in bytes) created in the last instantiation of this work area. This column is null if the last instantiation of this work area did not spill to disk. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

| Column | Datatype | Description |
|---|---|---|
| CON_DBID | NUMBER | The database ID of the PDB |

**See Also:**

- "V$SQL_PLAN_STATISTICS"
- "V$SQL_WORKAREA"

# V$SQL_REDIRECTION

V$SQL_REDIRECTION displays SQL statements that are redirected.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the cursor handle |
| PARENT_HANDLE | RAW(4 \| 8) | Address of the parent cursor handle |
| HASH_VALUE | NUMBER | Hash value of the SQL statement |
| SQL_ID | VARCHAR2(13) | SQL identifier of the SQL statement |
| CHILD_NUMBER | NUMBER | Number of the child (instance) for the hash |
| PARSING_USER_ID | NUMBER | Parsing user ID |
| PARSING_SCHEMA_ID | NUMBER | Parsing schema ID |
| COMMAND_TYPE | NUMBER | SELECT, UPDATE, INSERT, MERGE |
| REASON | VARCHAR2(14) | Reason for redirection ('INVALID OBJECT', 'ROWID', 'QUERY REWRITE', 'READ ONLY') |
| ERROR_CODE | NUMBER | Error code for local parse |
| POSITION | NUMBER | Error position, 0 if unknown |
| SQL_TEXT_PIECE | VARCHAR2(1000) | SQL Text containing position, usually a (qualified) identifier |
| ERROR_MESSAGE | VARCHAR2(1000) | Error code's corresponding error message resolved in the database language, no arguments resolved |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_SHARED_CURSOR

V$SQL_SHARED_CURSOR explains why a particular child cursor is not shared with existing child cursors. Each column identifies a specific reason why the cursor cannot be shared.

| Column | Datatype | Description |
|---|---|---|
| SQL_ID | VARCHAR2(13) | SQL identifier |
| ADDRESS | RAW(4 \| 8) | Address of the parent cursor |
| CHILD_ADDRESS | RAW(4 \| 8) | Address of the child cursor |
| CHILD_NUMBER | NUMBER | Child number |
| UNBOUND_CURSOR | VARCHAR2(1) | (Y\|N) The existing child cursor was not fully built (in other words, it was not optimized) |
| SQL_TYPE_MISMATCH | VARCHAR2(1) | (Y\|N) The SQL type does not match the existing child cursor |
| OPTIMIZER_MISMATCH | VARCHAR2(1) | (Y\|N) The optimizer environment does not match the existing child cursor |
| OUTLINE_MISMATCH | VARCHAR2(1) | (Y\|N) The outlines do not match the existing child cursor |
| STATS_ROW_MISMATCH | VARCHAR2(1) | (Y\|N) The existing statistics do not match the existing child cursor |
| LITERAL_MISMATCH | VARCHAR2(1) | (Y\|N) Non-data literal values do not match the existing child cursor |
| FORCE_HARD_PARSE | VARCHAR2(1) | (Y\|N) For internal use |
| EXPLAIN_PLAN_CURSOR | VARCHAR2(1) | (Y\|N) The child cursor is an explain plan cursor and should not be shared |
| BUFFERED_DML_MISMATCH | VARCHAR2(1) | (Y\|N) Buffered DML does not match the existing child cursor |
| PDML_ENV_MISMATCH | VARCHAR2(1) | (Y\|N) PDML environment does not match the existing child cursor |
| INST_DRTLD_MISMATCH | VARCHAR2(1) | (Y\|N) Insert direct load does not match the existing child cursor |
| SLAVE_QC_MISMATCH | VARCHAR2(1) | (Y\|N) The existing child cursor is a slave cursor and the new one was issued by the coordinator (or, the existing child cursor was issued by the coordinator and the new one is a slave cursor) |
| TYPECHECK_MISMATCH | VARCHAR2(1) | (Y\|N) The existing child cursor is not fully optimized |
| AUTH_CHECK_MISMATCH | VARCHAR2(1) | (Y\|N) Authorization/translation check failed for the existing child cursor |
| BIND_MISMATCH | VARCHAR2(1) | (Y\|N) The bind metadata does not match the existing child cursor |
| DESCRIBE_MISMATCH | VARCHAR2(1) | (Y\|N) The typecheck heap is not present during the describe for the child cursor |

| Column | Datatype | Description |
|---|---|---|
| LANGUAGE_MISMATCH | VARCHAR2(1) | (Y\|N) The language handle does not match the existing child cursor |
| TRANSLATION_MISMATCH | VARCHAR2(1) | (Y\|N) The base objects of the existing child cursor do not match |
| BIND_EQUIV_FAILURE | VARCHAR2(1) | (Y\|N) The bind value's selectivity does not match that used to optimize the existing child cursor |
| INSUFF_PRIVS | VARCHAR2(1) | (Y\|N) Insufficient privileges on objects referenced by the existing child cursor |
| INSUFF_PRIVS_REM | VARCHAR2(1) | (Y\|N) Insufficient privileges on remote objects referenced by the existing child cursor |
| REMOTE_TRANS_MISMATCH | VARCHAR2(1) | (Y\|N) The remote base objects of the existing child cursor do not match |
| LOGMINER_SESSION_MISMATCH | VARCHAR2(1) | (Y\|N) LogMiner Session parameters mismatch |
| INCOMP_LTRL_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor might have some binds (literals) which may be unsafe/non-data. Value mismatch. |
| OVERLAP_TIME_MISMATCH | VARCHAR2(1) | (Y\|N) Mismatch caused by setting session parameter ERROR_ON_OVERLAP_TIME |
| EDITION_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor edition mismatch |
| MV_QUERY_GEN_MISMATCH | VARCHAR2(1) | (Y\|N) Internal, used to force a hard-parse when analyzing materialized view queries |
| USER_BIND_PEEK_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor is not shared because value of one or more user binds is different and this has a potential to change the execution plan |
| TYPCHK_DEP_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor has typecheck dependencies |
| NO_TRIGGER_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor and child have no trigger mismatch |
| FLASHBACK_CURSOR | VARCHAR2(1) | (Y\|N) Cursor non-shareability due to flashback |
| ANYDATA_TRANSFORMATION | VARCHAR2(1) | (Y\|N) Is criteria for opaque type transformation and does not match |
| PDDL_ENV_MISMATCH | VARCHAR2(1) | (Y\|N) Environment setting mismatch for parallel DDL cursor (that is, one or more of the following parameter values have changed: PARALLEL_EXECUTION_ENABLED, PARALLEL_DDL_MODE, PARALLEL_DDL_FORCED_DEGREE, or PARALLEL_DDL_FORCED_INSTANCES) |
| TOP_LEVEL_RPI_CURSOR | VARCHAR2(1) | (Y\|N) Is top level RPI cursor |
| DIFFERENT_LONG_LENGTH | VARCHAR2(1) | (Y\|N) Value of LONG does not match |

| Column | Datatype | Description |
|---|---|---|
| LOGICAL_STANDBY_APPLY | VARCHAR2(1) | (Y\|N) Logical standby apply context does not match |
| DIFF_CALL_DURN | VARCHAR2(1) | (Y\|N) If Slave SQL cursor/single call |
| BIND_UACS_DIFF | VARCHAR2(1) | (Y\|N) One cursor has bind UACs and one does not |
| PLSQL_CMP_SWITCHS_DIFF | VARCHAR2(1) | (Y\|N) PL/SQL anonymous block compiled with different PL/SQL compiler switches |
| CURSOR_PARTS_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor was compiled with subexecution (cursor parts were executed) |
| STB_OBJECT_MISMATCH | VARCHAR2(1) | (Y\|N) STB is an internal name for a SQL Management Object Mismatch. A SQL Management Object Mismatch means that either a SQL plan baseline, or a SQL profile, or a SQL patch has been created for your SQL statement between the executions. Because a cursor is a read-only entity, a hard parse is forced to be able to create a new cursor that contains information about the new SQL management object related to this SQL statement. |
| CROSSEDITION_TRIGGER_MISMATCH | VARCHAR2(1) | (Y\|N) The set of crossedition triggers to execute might differ |
| PQ_SLAVE_MISMATCH | VARCHAR2(1) | (Y\|N) Top-level slave decides not to share cursor |
| TOP_LEVEL_DDL_MISMATCH | VARCHAR2(1) | (Y\|N) Is top-level DDL cursor |
| MULTI_PX_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor has multiple parallelizers and is slave-compiled |
| BIND_PEEKED_PQ_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor based around bind peeked values |
| MV_REWRITE_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor needs recompilation because an SCN was used during compile time due to being rewritten by materialized view |
| ROLL_INVALID_MISMATCH | VARCHAR2(1) | (Y\|N) Marked for rolling invalidation and invalidation window exceeded |
| OPTIMIZER_MODE_MISMATCH | VARCHAR2(1) | (Y\|N) Parameter OPTIMIZER_MODE mismatch (for example, all_rows versus first_rows_1) |
| PX_MISMATCH | VARCHAR2(1) | (Y\|N) Mismatch in one parameter affecting the parallelization of a SQL statement. For example, one cursor was compiled with parallel DML enabled while the other was not. |
| MV_STALEOBJ_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor cannot be shared because there is a mismatch in the list of materialized views which were stale when the cursor was built |
| FLASHBACK_TABLE_MISMATCH | VARCHAR2(1) | (Y\|N) Cursor cannot be shared because there is a mismatch with triggers being enabled and/or referential integrity constraints being deferred |
| LITREP_COMP_MISMATCH | VARCHAR2(1) | (Y\|N) Mismatch in use of literal replacement |
| PLSQL_DEBUG | VARCHAR2(1) | (Y\|N) Value of the PLSQL_DEBUG parameter for the current session does not match the value used to build the cursor |

| Column | Datatype | Description |
|---|---|---|
| LOAD_OPTIMIZER_STATS | VARCHAR2(1) | (Y\|N) A hard parse is forced to initialize extended cursor sharing |
| ACL_MISMATCH | VARCHAR2(1) | (Y\|N) Cached ACL evaluation result stored in the child cursor is not valid for the current session or user |
| FLASHBACK_ARCHIVE_MISMATCH | VARCHAR2(1) | (Y\|N) Value of the FLASHBACK_DATA_ARCHIVE_INTERNAL_CURSOR parameter for the current session does not match the value used to build the cursor |
| LOCK_USER_SCHEMA_FAILED | VARCHAR2(1) | (Y\|N) User or schema used to build the cursor no longer exists<br>**Note:** This sharing criterion is deprecated |
| REMOTE_MAPPING_MISMATCH | VARCHAR2(1) | (Y\|N) Reloaded cursor was previously remote-mapped and is currently not remote-mapped. Therefore, the cursor must be reparsed. |
| LOAD_RUNTIME_HEAP_FAILED | VARCHAR2(1) | (Y\|N) Loading of run-time heap for the new cursor (or reload of aged out cursor) failed |
| HASH_MATCH_FAILED | VARCHAR2(1) | (Y\|N) No existing child cursors have the unsafe literal bind hash values required by the current cursor |
| PURGED_CURSOR | VARCHAR2(1) | (Y\|N) Child cursor is marked for purging |
| BIND_LENGTH_UPGRADEABLE | VARCHAR2(1) | (Y\|N) Bind length(s) required for the current cursor are longer than the bind length(s) used to build the child cursor |
| USE_FEEDBACK_STATS | VARCHAR2(1) | (Y\|N) A hard parse is forced so that the optimizer can reoptimize the query with improved optimizer inputs (for example, cardinality estimates) |
| REASON | CLOB | Child number, id, and reason the cursor is not shared. The content of this column is structured using XML. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_SHARED_MEMORY

V$SQL_SHARED_MEMORY displays information about the cursor shared memory snapshot. Each SQL statement stored in the shared pool has one or more child objects associated with it. Each child object has several parts, one of which is the context heap, which holds, among other things, the query plan.

| Column | Datatype | Description |
|---|---|---|
| SQL_TEXT | VARCHAR2(1000) | SQL text of the shared cursor child object for which this row is displaying information |

| Column | Datatype | Description |
|---|---|---|
| SQL_FULLTEXT | CLOB | Full text for the SQL statement exposed as a CLOB column. The full text of a SQL statement can be retrieved using this column instead of joining with the V$SQLTEXT dynamic performance view. |
| HASH_VALUE | NUMBER | Hash value of the above SQL text in the shared pool |
| SQL_ID | VARCHAR2(13) | SQL identifier of the above SQL text in the shared pool |
| HEAP_DESC | RAW(4 \| 8) | Address of the descriptor for the context heap of the child cursor described in this row |
| STRUCTURE | VARCHAR2(16) | If the memory chunk described in this row was allocated using a comment of the form "X : Y", then this is the "X" part of the comment |
| FUNCTION | VARCHAR2(16) | Similar to the STRUCTURE column, this is the "Y" field of the comment |
| CHUNK_COM | VARCHAR2(16) | Whole comment field that was supplied when this memory chunk was allocated |
| CHUNK_PTR | RAW(4 \| 8) | Starting address of the allocated memory chunk |
| CHUNK_SIZE | NUMBER | Amount of memory allocated for this chunk |
| ALLOC_CLASS | VARCHAR2(8) | Class of memory that this chunk of memory belongs to. It will usually be either FREEABLE or PERMANENT. |
| CHUNK_TYPE | NUMBER | An index into a table of callback functions that tell the server how to re-create this chunk of memory should it need to be removed from the shared pool based on an LRU algorithm |
| SUBHEAP_DESC | RAW(4 \| 8) | If the parent heap of this context heap is itself a subheap, then this is the address of the descriptor of the parent heap |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQL_WORKAREA

V$SQL_WORKAREA displays information about work areas used by SQL cursors. Each SQL statement stored in the shared pool has one or more child cursors that are listed in the V$SQL view. V$SQL_WORKAREA lists all work areas needed by these child cursors; V$SQL_WORKAREA can be joined with V$SQLAREA on (ADDRESS, HASH_VALUE) and with V$SQL on (ADDRESS, HASH_VALUE, CHILD_NUMBER).

You can use this view to find out answers to the following questions:

- What are the top 10 work areas that require the most cache area?

- For work areas allocated in AUTO mode, what percentage of work areas are running using maximum memory?

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Address of the parent cursor handle |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache. Two columns PARENT_HANDLE and HASH_VALUE can be used to join with V$SQLAREA to locate the parent cursor. |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent statement in the library cache |
| CHILD_NUMBER | NUMBER | Number of the child cursor that uses this work area. The columns PARENT_HANDLE, HASH_VALUE, and CHILD_NUMBER can be used to join with V$SQL to locate the child cursor using this area. |
| WORKAREA_ADDRESS | RAW(4 \| 8) | Address of the work area handle. This is the primary key for the view. |
| OPERATION_TYPE | VARCHAR2(40) | Type of operation using the work area. Can include values such as SORT, HASH JOIN, GROUP BY, BUFFER, BITMAP MERGE, and BITMAP CONSTRUCTION. |
| OPERATION_ID | NUMBER | A unique number used to identify the operation in the execution plan. This identifier can be joined to V$SQL_PLAN to locate the operation that uses this work area. |
| POLICY | VARCHAR2(10) | Sizing policy for this work area (MANUAL or AUTO) |
| ESTIMATED_OPTIMAL_SIZE | NUMBER | Estimated size (in bytes) required by this work area to execute the operation completely in memory (optimal execution). Derived from either optimizer statistics or previous executions. |
| ESTIMATED_ONEPASS_SIZE | NUMBER | Estimated size (in bytes) required by this work area to execute the operation in a single pass. Derived from either optimizer statistics or previous executions. |
| LAST_MEMORY_USED | NUMBER | Memory (in bytes) used by this work area during the last execution of the cursor |
| LAST_EXECUTION | VARCHAR2(10) | Indicates whether this work area runs using OPTIMAL, ONE PASS, or ONE PASS memory requirement (or MULTI-PASS), during the last execution of the cursor |
| LAST_DEGREE | NUMBER | Degree of parallelism used during the last execution of this operation |
| TOTAL_EXECUTIONS | NUMBER | Number of times this work area was active |
| OPTIMAL_EXECUTIONS | NUMBER | Number of times this work area ran in optimal mode |
| ONEPASS_EXECUTIONS | NUMBER | Number of times this work area ran in one-pass mode |
| MULTIPASSES_EXECUTIONS | NUMBER | Number of times this work area ran below the one-pass memory requirement |
| ACTIVE_TIME | NUMBER | Average time this work area is active (in hundredths of a second) |
| MAX_TEMPSEG_SIZE | NUMBER | Maximum temporary segment size (in bytes) created by an instantiation of this work area. This column is NULL if this work area has never spilled to disk. |

| Column | Datatype | Description |
|---|---|---|
| LAST_TEMPSEG_SIZE | NUMBER | Temporary segment size (in bytes) created in the last instantiation of this work area. This column is NULL if the last instantiation of this work area did not spill to disk. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SQLAREA"
- "V$SQL"

# V$SQL_WORKAREA_ACTIVE

V$SQL_WORKAREA_ACTIVE contains an instantaneous view of the work areas currently allocated by the system. You can join this view against V$SQL_WORKAREA on WORKAREA_ADDRESS to access the definition of that work area. If a work area spills to disk, then this view contains information for the temporary segment created on behalf of this work area.

The last three columns are included to enable joining V$SQL_WORKAREA_ACTIVE with V$TEMPSEG_USAGE to retrieve more information on this temporary segment.

You can use this view to answer the following:

- What are the top 10 largest work areas currently allocated in the system?
- What percentage of memory is over-allocated (EXPECTED_SIZE < ACTUAL_MEM_USED) and under-allocated (EXPECTED_SIZE > ACTUAL_MEM_USED)?
- What are the active work areas using more memory than what is expected by the memory manager?
- What are the active work areas that have spilled to disk?

| Column | Datatype | Description |
|---|---|---|
| SQL_HASH_VALUE | NUMBER | Hash value of the SQL statement that is currently being executed |
| SQL_ID | VARCHAR2(13) | SQL identifier of the SQL statement that is currently being executed |
| SQL_EXEC_START | DATE | Time when the execution of the SQL currently executed by this session started |
| SQL_EXEC_ID | NUMBER | SQL execution identifier (see V$SQL_MONITOR) |

| Column | Datatype | Description |
|---|---|---|
| WORKAREA_ADDRESS | RAW(4 \| 8) | Address of the work area handle. This is the primary key for the view. |
| OPERATION_TYPE | VARCHAR2(40) | Type of operation using the work area. Can include values such as SORT, HASH JOIN, GROUP BY, BUFFER, BITMAP MERGE, and BITMAP CONSTRUCTION. |
| OPERATION_ID | NUMBER | A unique number used to identify the operation in the execution plan. This identifier can be joined to V$SQL_PLAN to locate the operation that uses this work area. |
| POLICY | VARCHAR2(6) | Sizing policy for this work area (MANUAL or AUTO) |
| SID | NUMBER | Session identifier |
| QCINST_ID | NUMBER | Query coordinator instance identifier. Along with QCSID, enables you to uniquely identify the query coordinator. |
| QCSID | NUMBER | Query coordinator session identifier. This is the same as the SID if the work area is allocated by a serial cursor. |
| ACTIVE_TIME | NUMBER | Average time this work area is active (in microseconds) |
| WORK_AREA_SIZE | NUMBER | Maximum size (in bytes) of the work area as it is currently used by the operation |
| EXPECTED_SIZE | NUMBER | Expected size (in bytes) for this work area. EXPECTED_SIZE is set on behalf of the operation by the memory manager. Memory can be over-allocated when WORK_AREA_SIZE has a higher value than EXPECTED_SIZE. This can occur when the operation using this work area takes a long time to resize it. |
| ACTUAL_MEM_USED | NUMBER | Amount of PGA memory (in bytes) currently allocated on behalf of this work area. This value should range between 0 and WORK_AREA_SIZE. |
| MAX_MEM_USED | NUMBER | Maximum memory amount (in bytes) used by this work area |
| NUMBER_PASSES | NUMBER | Number of passes corresponding to this work area (0 if running in OPTIMAL mode) |
| TEMPSEG_SIZE | NUMBER | Size (in bytes) of the temporary segment used on behalf of this work area. This column is NULL if this work area has not (yet) spilled to disk. |
| TABLESPACE | VARCHAR2(30) | Tablespace name for the temporary segment created on behalf of this work area. This column is NULL if this work area has not (yet) spilled to disk. |
| SEGRFNO# | NUMBER | Relative file number within the tablespace for the temporary segment created on behalf of this work area. This column is NULL if this work area has not (yet) spilled to disk. |
| SEGBLK# | NUMBER | Block number for the temporary segment created on behalf of this work area. This column is NULL if this work area has not (yet) spilled to disk. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SQL_WORKAREA"
- *Oracle Database Concepts* for more information about SQL work areas

# V$SQL_WORKAREA_HISTOGRAM

V$SQL_WORKAREA_HISTOGRAM displays the cumulative work area execution statistics (cumulated since instance startup) for different work area groups. The work areas are split into 33 groups based on their optimal memory requirements with the requirements increasing in powers of two. That is, work areas whose optimal requirement varies from 0 KB to 1 KB, 1 KB to 2 KB, 2 KB to 4 KB, ... and 2 TB to 4 TB.

For each work area group, the V$SQL_WORKAREA_HISTOGRAM view shows how many work areas in that group were able to run in optimal mode, how many were able to run in one-pass mode, and finally how many ran in multi-pass mode. The DBA can take a snapshot at the beginning and the end of a desired time interval to derive the same statistics for that interval.

| Column | Datatype | Description |
|---|---|---|
| LOW_OPTIMAL_SIZE | NUMBER | Lower bound for the optimal memory requirement of work areas included in this row (bytes) |
| HIGH_OPTIMAL_SIZE | NUMBER | Upper bound for the optimal memory requirement of work areas included in this row (bytes) |
| OPTIMAL_EXECUTIONS | NUMBER | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which have been executed in optimal mode since instance startup |
| ONEPASS_EXECUTIONS | NUMBER | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which have been executed in one-pass mode since instance startup |
| MULTIPASSES_EXECUTIONS | NUMBER | Number of work areas with an optimal memory requirement comprised between LOW_OPTIMAL_SIZE and HIGH_OPTIMAL_SIZE which have been executed in multi-pass mode since instance startup |
| TOTAL_EXECUTIONS | NUMBER | Sum of OPTIMAL_EXECUTIONS, ONEPASS_EXECUTIONS, and MULTIPASSES_EXECUTIONS |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Performance Tuning Guide* for detailed information on how to monitor automatic PGA memory performance using this view

# V$SQLAREA

V$SQLAREA displays statistics on shared SQL areas and contains one row per SQL string. It provides statistics on SQL statements that are in memory, parsed, and ready for execution.

| Column | Datatype | Description |
|---|---|---|
| SQL_TEXT | VARCHAR2(1000) | First thousand characters of the SQL text for the current cursor |
| SQL_FULLTEXT | CLOB | All characters of the SQL text for the current cursor |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| SHARABLE_MEM | NUMBER | Amount of shared memory used by a cursor. If multiple child cursors exist, then the sum of all shared memory used by all child cursors. |
| PERSISTENT_MEM | NUMBER | Fixed amount of memory used for the lifetime of an open cursor. If multiple child cursors exist, then the fixed sum of memory used for the lifetime of all the child cursors. |
| RUNTIME_MEM | NUMBER | Fixed amount of memory required during execution of a cursor. If multiple child cursors exist, then the fixed sum of all memory required during execution of all the child cursors. |
| SORTS | NUMBER | Sum of the number of sorts that were done for all the child cursors |
| VERSION_COUNT | NUMBER | Number of child cursors that are present in the cache under this parent |
| LOADED_VERSIONS | NUMBER | Number of child cursors that are present in the cache and have their context heap loaded |
| OPEN_VERSIONS | NUMBER | Number of child cursors that are currently open under this current parent |
| USERS_OPENING | NUMBER | Number of users that have any of the child cursors open |
| FETCHES | NUMBER | Number of fetches associated with the SQL statement |
| EXECUTIONS | NUMBER | Total number of executions, totalled over all the child cursors |

| Column | Datatype | Description |
|---|---|---|
| PX_SERVERS_EXECUTIONS | NUMBER | Total number of executions performed by parallel execution servers (0 when the statement has never been executed in parallel) |
| END_OF_FETCH_COUNT | NUMBER | Number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less or equal to the value of the EXECUTIONS column. |
| USERS_EXECUTING | NUMBER | Total number of users executing the statement over all child cursors |
| LOADS | NUMBER | Number of times the object was loaded or reloaded |
| FIRST_LOAD_TIME | VARCHAR2(19) | Timestamp of the parent creation time |
| INVALIDATIONS | NUMBER | Total number of invalidations over all the child cursors |
| PARSE_CALLS | NUMBER | Sum of all parse calls to all the child cursors under this parent |
| DISK_READS | NUMBER | Sum of the number of disk reads over all child cursors |
| DIRECT_WRITES | NUMBER | Sum of the number of direct writes over all child cursors |
| BUFFER_GETS | NUMBER | Sum of buffer gets over all child cursors |
| APPLICATION_WAIT_TIME | NUMBER | Application wait time (in microseconds) |
| CONCURRENCY_WAIT_TIME | NUMBER | Concurrency wait time (in microseconds) |
| CLUSTER_WAIT_TIME | NUMBER | Cluster wait time (in microseconds) |
| USER_IO_WAIT_TIME | NUMBER | User I/O Wait Time (in microseconds) |
| PLSQL_EXEC_TIME | NUMBER | PL/SQL execution time (in microseconds) |
| JAVA_EXEC_TIME | NUMBER | Java execution time (in microseconds) |
| ROWS_PROCESSED | NUMBER | Total number of rows processed on behalf of this SQL statement |
| COMMAND_TYPE | NUMBER | Oracle command type definition |
| OPTIMIZER_MODE | VARCHAR2(10) | Mode under which the SQL statement was executed |
| OPTIMIZER_COST | NUMBER | Cost of this query given by the optimizer |
| OPTIMIZER_ENV | RAW(2000) | Optimizer environment |
| OPTIMIZER_ENV_HASH_VALUE | NUMBER | Hash value for the optimizer environment |
| PARSING_USER_ID | NUMBER | User ID of the user that has parsed the very first cursor under this parent |

| Column | Datatype | Description |
|---|---|---|
| PARSING_SCHEMA_ID | NUMBER | Schema ID that was used to parse this child cursor |
| PARSING_SCHEMA_NAME | VARCHAR2(30) | Schema name that was used to parse this child cursor |
| KEPT_VERSIONS | NUMBER | Number of child cursors that have been marked to be kept using the DBMS_SHARED_POOL package |
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache |
| OLD_HASH_VALUE | NUMBER | Old SQL hash value |
| PLAN_HASH_VALUE | NUMBER | Numeric representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). |
| FULL_PLAN_HASH_VALUE | NUMBER | Numeric representation of the complete SQL plan for this cursor. Comparing one FULL_PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). Note that the FULL_PLAN_HASH_VALUE cannot be compared across databases releases. It is not backward compatible. |
| MODULE | VARCHAR2(64) | Contains the name of the module that was executing when the SQL statement was first parsed as set by calling DBMS_APPLICATION_INFO.SET_MODULE |
| MODULE_HASH | NUMBER | Hash value of the module that is named in the MODULE column |
| ACTION | VARCHAR2(64) | Contains the name of the action that was executing when the SQL statement was first parsed as set by calling DBMS_APPLICATION_INFO.SET_ACTION |
| ACTION_HASH | NUMBER | Hash value of the action that is named in the ACTION column |
| SERIALIZABLE_ABORTS | NUMBER | Number of times the transaction failed to serialize, producing ORA-08177 errors, totalled over all the child cursors |
| OUTLINE_CATEGORY | VARCHAR2(64) | If an outline was applied during construction of the cursor, then this column displays the category of that outline. Otherwise the column is left blank. |
| CPU_TIME | NUMBER | CPU time (in microseconds) used by this cursor for parsing, executing, and fetching |
| ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) used by this cursor for parsing, executing, and fetching. If the cursor uses parallel execution, then ELAPSED_TIME is the cumulative time for the query coordinator, plus all parallel query slave processes. |
| OUTLINE_SID | VARCHAR2(40) | Outline session identifier |
| LAST_ACTIVE_CHILD_ADDRESS | RAW(4 \| 8) | Address (identifier) of the child cursor that was the last to be active in the group (that is, the child cursor on behalf of which statistics in V$SQL were updated) |
| REMOTE | VARCHAR2(1) | Indicates whether the cursor is remote mapped (Y) or not (N) |

| Column | Datatype | Description |
|---|---|---|
| OBJECT_STATUS | VARCHAR2(19) | Status of the cursor:<br>• VALID - Valid, authorized without errors<br>• VALID_AUTH_ERROR - Valid, authorized with authorization errors<br>• VALID_COMPILE_ERROR - Valid, authorized with compilation errors<br>• VALID_UNAUTH - Valid, unauthorized<br>• INVALID_UNAUTH - Invalid, unauthorized<br>• INVALID - Invalid, unauthorized but keep the timestamp |
| LITERAL_HASH_VALUE | NUMBER | Hash value of the literals which are replaced with system-generated bind variables and are to be matched, when CURSOR_SHARING is used. This is not the hash value for the SQL statement. If CURSOR_SHARING is not used, then the value is 0. |
| LAST_LOAD_TIME | DATE | Time at which the query plan was loaded into the library cache |
| IS_OBSOLETE | VARCHAR2(1) | Indicates whether the cursor has become obsolete (Y) or not (N). This can happen if the number of child cursors is too large. |
| IS_BIND_SENSITIVE | VARCHAR2(1) | Indicates whether the cursor is bind sensitive (Y) or not (N). A query is considered bind-sensitive if the optimizer peeked at one of its bind variable values when computing predicate selectivities and where a change in a bind variable value may cause the optimizer to generate a different plan. |
| IS_BIND_AWARE | VARCHAR2(1) | Indicates whether the cursor is bind aware (Y) or not (N). A query is considered bind-aware if it has been marked to use extended cursor sharing. The query would already have been marked as bind-sensitive. |
| CHILD_LATCH | NUMBER | Child latch number that is protecting the cursor. This column is obsolete and maintained for backward compatibility. |
| SQL_PROFILE | VARCHAR2(64) | SQL profile used for this statement, if any |
| SQL_PATCH | VARCHAR2(30) | SQL patch used for this statement, if any |
| SQL_PLAN_BASELINE | VARCHAR2(30) | SQL plan baseline used for this statement, if any |
| PROGRAM_ID | NUMBER | Program identifier |
| PROGRAM_LINE# | NUMBER | Program line number |
| EXACT_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to EXACT |
| FORCE_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to FORCE |
| LAST_ACTIVE_TIME | DATE | Time at which the query plan was last active |
| BIND_DATA | RAW(2000) | Bind data |
| TYPECHECK_MEM | NUMBER | Typecheck memory |
| IO_CELL_OFFLOAD_ELIGIBLE_BYTES | NUMBER | Number of I/O bytes which can be filtered by the Exadata storage system |

| Column | Datatype | Description |
|---|---|---|
| | | **See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_INTERCONNECT_BYTES | NUMBER | Number of I/O bytes exchanged between Oracle Database and the storage system |
| PHYSICAL_READ_REQUESTS | NUMBER | Number of physical read I/O requests issued by the monitored SQL |
| PHYSICAL_READ_BYTES | NUMBER | Number of bytes read from disks by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS | NUMBER | Number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_BYTES | NUMBER | Number of bytes written to disks by the monitored SQL |
| OPTIMIZED_PHY_READ_REQUESTS | NUMBER | Number of physical read I/O requests from Database Smart Flash Cache issued by the monitored SQL |
| LOCKED_TOTAL | NUMBER | Total number of times the child cursor has been locked |
| PINNED_TOTAL | NUMBER | Total number of times the child cursor has been pinned |
| IO_CELL_UNCOMPRESSED_BYTES | NUMBER | Number of uncompressed bytes (that is, size after decompression) that are offloaded to the Exadata cells<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_CELL_OFFLOAD_RETURNED_BYTES | NUMBER | Number of bytes that are returned by Exadata cell through the regular I/O path<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| IS_REOPTIMIZABLE | VARCHAR2(1) | This columns shows whether the next execution matching this child cursor will trigger a reoptimization. The values are:<br>• `Y`: If the next execution will trigger a reoptimization<br>• `R`: If the child cursor contains reoptimization information, but will not trigger reoptimization because the cursor was compiled in reporting mode<br>• `N`: If the child cursor has no reoptimization information |
| IS_RESOLVED_ADAPTIVE_PLAN | VARCHAR2(1) | This column shows whether all of the adaptive parts of a plan have been resolved to the final plan. Once the plan is resolved, the plan hash value and the plan displayed by `DBMS_XPLAN` will not change through the end of execution. The values for this column are:<br>• `NULL`: If the plan is not adaptive |

| Column | Datatype | Description |
|---|---|---|
| | | • Y: If the plan is fully resolved<br>• N: If the plan is not yet fully resolved<br>**See Also:** *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XPLAN package |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SHARED_POOL package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_MODULE procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_ACTION procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XPLAN package

# V$SQLAREA_PLAN_HASH

V$SQLAREA_PLAN_HASH displays statistics on shared SQL areas (V$SQL) by grouping on the SQL_ID and PLAN_HASH_VALUE columns. It can potentially create several rows for one parent cursor, one for each distinct value of the column PLAN_HASH_VALUE.

| Column | Datatype | Description |
|---|---|---|
| SQL_TEXT | VARCHAR2(1000) | First thousand characters of the SQL text for the current cursor |
| SQL_FULLTEXT | CLOB | All characters of the SQL text for the current cursor |
| ADDRESS | RAW(4 \| 8) | Address of the handle to the parent for this cursor |
| HASH_VALUE | NUMBER | Hash value of the parent statement in the library cache |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| PLAN_HASH_VALUE | NUMBER | Numeric representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). |
| VERSION_COUNT | NUMBER | Number of child cursors that are present in the cache under this parent |
| LAST_ACTIVE_CHILD_ADDRESS | RAW(4 \| 8) | Address (identifier) of the child cursor that was the last to be active in the group (that is, the child cursor on behalf of which statistics in V$SQL were updated) |
| SHARABLE_MEM | NUMBER | Amount of shared memory used by a cursor. If multiple child cursors exist, then it is the sum of all shared memory used by all child cursors. |

| Column | Datatype | Description |
|---|---|---|
| PERSISTENT_MEM | NUMBER | Fixed amount of memory used for the lifetime of an open cursor. If multiple child cursors exist, then it is the fixed sum of memory used for the lifetime of all the child cursors. |
| RUNTIME_MEM | NUMBER | Fixed amount of memory required during execution of a cursor. If multiple child cursors exist, then the fixed sum of all memory required during execution of all the child cursors. |
| SORTS | NUMBER | Sum of the number of sorts that were done for all the child cursors |
| LOADED_VERSIONS | NUMBER | Number of child cursors that are present in the cache and that have their context heap loaded |
| OPEN_VERSIONS | NUMBER | Number of child cursors that are currently open under this parent |
| USERS_OPENING | NUMBER | Number of users that have any of the child cursors open |
| USERS_EXECUTING | NUMBER | Total number of users executing the statement over all child cursors |
| FETCHES | NUMBER | Number of fetches associated with the SQL statement |
| EXECUTIONS | NUMBER | Total number of executions, totalled over all the child cursors |
| PX_SERVERS_EXECUTIONS | NUMBER | Total number of executions performed by parallel execution servers (0 when the statement has never been executed in parallel) |
| END_OF_FETCH_COUNT | NUMBER | Number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less or equal to the value of the EXECUTIONS column. |
| LOADS | NUMBER | Number of times the object was loaded or reloaded |
| FIRST_LOAD_TIME | DATE | Timestamp of the parent creation time |
| LAST_LOAD_TIME | DATE | Time at which the query plan was loaded into the library cache |
| LAST_ACTIVE_TIME | DATE | TIme at which the query plan was last active |
| INVALIDATIONS | NUMBER | Total number of invalidations over all the child cursors |
| PARSE_CALLS | NUMBER | Sum of all parse calls to all the child cursors under this parent |
| DISK_READS | NUMBER | Sum of the number of disk reads over all child cursors |
| DIRECT_WRITES | NUMBER | Sum of the number of direct writes over all child cursors |
| BUFFER_GETS | NUMBER | Sum of buffer gets over all child cursors |
| CPU_TIME | NUMBER | CPU time (in microseconds) used by this cursor for parsing, executing, and fetching |

| Column | Datatype | Description |
|---|---|---|
| ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) used by this cursor for parsing, executing, and fetching |
| APPLICATION_WAIT_TIME | NUMBER | Application wait time (in microseconds) |
| CONCURRENCY_WAIT_TIME | NUMBER | Concurrency wait time (in microseconds) |
| CLUSTER_WAIT_TIME | NUMBER | Cluster wait time (in microseconds) |
| USER_IO_WAIT_TIME | NUMBER | User I/O wait time (in microseconds) |
| PLSQL_EXEC_TIME | NUMBER | PL/SQL execution time (in microseconds) |
| JAVA_EXEC_TIME | NUMBER | Java execution time (in microseconds) |
| ROWS_PROCESSED | NUMBER | Total number of rows processed on behalf of this SQL statement |
| COMMAND_TYPE | NUMBER | Oracle command type definition |
| OPTIMIZER_MODE | VARCHAR2(10) | Mode under which the SQL statement was executed |
| OPTIMIZER_COST | NUMBER | Cost of this query given by the optimizer |
| OPTIMIZER_ENV | RAW(2000) | Optimizer environment |
| OPTIMIZER_ENV_HASH_VALUE | NUMBER | Hash value for the optimizer environment |
| PARSING_USER_ID | NUMBER | User ID of the user that has parsed the very first cursor under this parent |
| PARSING_SCHEMA_ID | NUMBER | Schema ID that was used to parse this child cursor |
| PARSING_SCHEMA_NAME | VARCHAR2(30) | Schema name that was used to parse this child cursor |
| KEPT_VERSIONS | NUMBER | Number of child cursors that have been marked to be kept using the DBMS_SHARED_POOL package |
| MODULE | VARCHAR2(64) | Contains the name of the module that was executing when the SQL statement was first parsed as set by calling DBMS_APPLICATION_INFO.SET_MODULE |
| MODULE_HASH | NUMBER | Hash value of the module that is named in the MODULE column |
| ACTION | VARCHAR2(64) | Contains the name of the action that was executing when the SQL statement was first parsed as set by calling DBMS_APPLICATION_INFO.SET_ACTION |
| ACTION_HASH | NUMBER | Hash value of the action that is named in the ACTION column |
| SERIALIZABLE_ABORTS | NUMBER | Number of times the transaction failed to serialize, producing ORA-08177 errors, totalled over all the child cursors |
| OUTLINE_CATEGORY | VARCHAR2(64) | If an outline was applied during construction of the cursor, then this column displays the category of that outline. Otherwise, the column is left blank. |

| Column | Datatype | Description |
|---|---|---|
| OUTLINE_SID | VARCHAR2(40) | Outline session identifier |
| REMOTE | VARCHAR2(1) | Indicates whether the cursor is remote mapped (Y) or not (N) |
| OBJECT_STATUS | VARCHAR2(19) | Status of the cursor:<br>• VALID - Valid, authorized without errors<br>• VALID_AUTH_ERROR - Valid, authorized with authorization errors<br>• VALID_COMPILE_ERROR - Valid, authorized with compilation errors<br>• VALID_UNAUTH - Valid, unauthorized<br>• INVALID_UNAUTH - Invalid, unauthorized<br>• INVALID - Invalid, unauthorized but keep the timestamp |
| LITERAL_HASH_VALUE | NUMBER | Hash value of the literals which are replaced with system-generated bind variables and are to be matched, when CURSOR_SHARING is used. This is not the hash value for the SQL statement. If CURSOR_SHARING is not used, then the value is 0. |
| SQL_PROFILE | VARCHAR2(64) | SQL profile used for this statement, if any |
| PROGRAM_ID | NUMBER | Program identifier |
| PROGRAM_LINE# | NUMBER | Program line number |
| EXACT_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to EXACT |
| FORCE_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to FORCE |
| BIND_DATA | RAW(2000) | Bind data |
| TYPECHECK_MEM | NUMBER | Typecheck memory |
| IO_CELL_OFFLOAD_ELIGIBLE_BYTES | NUMBER | Number of I/O bytes which can be filtered by the Exadata storage system<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_INTERCONNECT_BYTES | NUMBER | Number of I/O bytes exchanged between Oracle Database and the storage system |
| PHYSICAL_READ_REQUESTS | NUMBER | Number of physical read I/O requests issued by the monitored SQL |
| PHYSICAL_READ_BYTES | NUMBER | Number of bytes read from disks by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS | NUMBER | Number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_BYTES | NUMBER | Number of bytes written to disks by the monitored SQL |
| OPTIMIZED_PHY_READ_REQUESTS | NUMBER | Number of physical read I/O requests from Database Smart Flash Cache issued by the monitored SQL |

| Column | Datatype | Description |
|---|---|---|
| IO_CELL_UNCOMPRESSED_BYTES | NUMBER | Number of uncompressed bytes (that is, size after decompression) that are offloaded to the Exadata cells<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_CELL_OFFLOAD_RETURNED_BYTES | NUMBER | Number of bytes that are returned by Exadata cell through the regular I/O path<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CON_DBID | NUMBER | The database ID of the PDB |

**See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_SHARED_POOL package
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_MODULE procedure
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_APPLICATION_INFO.SET_ACTION procedure

# V$SQLCOMMAND

V$SQLCOMMAND displays the mapping between SQL opcodes and names.

| Column | Datatype | Description |
|---|---|---|
| COMMAND_TYPE | NUMBER | SQL command number |
| COMMAND_NAME | VARCHAR2(64) | SQL command name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQLFN_ARG_METADATA

V$SQLFN_ARG_METADATA contains metadata about function arguments. There is one row for each argument of every function found in V$SQLFN_METADATA. There are no rows for functions that do not have any arguments.

| Column | Datatype | Description |
|---|---|---|
| FUNC_ID | NUMBER | Internal function identification number. This column can be used to join with the V$SQLFN_METADATA view. |
| ARGNUM | NUMBER | Argument number |
| DATATYPE | VARCHAR2(8) | Data type of the argument. The value is NULL if this argument is not used. Otherwise, it can take values of any Oracle data type, family data type, or EXPR data type. |
| DESCR | VARCHAR2(30) | This column is reserved for future use. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SQLFN_METADATA"

# V$SQLFN_METADATA

V$SQLFN_METADATA contains metadata about operators and built-in functions. Note that this view does not contain information about arguments because the number of arguments will be different for various functions. Information about arguments is contained in V$SQLFN_ARG_METADATA, which can be joined with V$SQLFN_METADATA to get information about any function and its arguments.

| Column | Datatype | Description |
|---|---|---|
| FUNC_ID | NUMBER | Internal function identification number |
| NAME | VARCHAR2(30) | Name of the built-in function |
| MINARGS | NUMBER | Minimum number of arguments for the function |
| MAXARGS | NUMBER | Maximum number of arguments for the function |
| DATATYPE | VARCHAR2(8) | Return data type, which can take any Oracle data type values, data type family values, or ARG[n] data types |
| VERSION | VARCHAR2(12) | Minimum version of the database that has this function |

| Column | Datatype | Description |
|---|---|---|
| ANALYTIC | VARCHAR2(3) | Indicates whether the function is an analytic function (YES) or not (NO) |
| AGGREGATE | VARCHAR2(3) | Indicates whether the function is an aggregate function (YES) or not (NO) |
| OFFLOADABLE | VARCHAR2(3) | Indicates whether execution of the function can be offloaded to the Oracle Exadata Storage Server (YES) or not (NO)<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| DISP_TYPE | VARCHAR2(13) | Function display type:<br>• NORMAL<br>• ARITHMETIC<br>• PARENTHESIS<br>• REL-OP<br>• CASELIKE<br>• NOPARENTHESIS |
| USAGE | VARCHAR2(30) | A text explanation of how to use this function. The text is based on the syntax diagram for the function in the *Oracle Database SQL Language Reference*. |
| DESCR | VARCHAR2(4000) | Description of the function |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SQLFN_ARG_METADATA"

# V$SQLSTATS

V$SQLSTATS displays basic performance statistics for SQL cursors and contains one row per SQL statement (that is, one row per unique value of SQL_ID). The column definitions for columns in V$SQLSTATS are identical to those in the V$SQL and V$SQLAREA views. However, the V$SQLSTATS view differs from V$SQL and V$SQLAREA in that it is faster, more scalable, and has a greater data retention (the statistics may still appear in this view, even after the cursor has been aged out of the shared pool). Note that V$SQLSTATS contains a subset of columns that appear in V$SQL and V$SQLAREA.

| Column | Datatype | Description |
|---|---|---|
| SQL_TEXT | VARCHAR2(1000) | First thousand characters of the SQL text for the current cursor |

| Column | Datatype | Description |
|---|---|---|
| SQL_FULLTEXT | CLOB | Full text for the SQL statement exposed as a CLOB column. THe full text of a SQL statement can be retrieved using this column instead of joining with the V$SQLTEXT view. |
| SQL_ID | VARCHAR2(13) | SQL identifier of the parent cursor in the library cache |
| LAST_ACTIVE_TIME | DATE | Last time the statistics of a contributing cursor were updated |
| LAST_ACTIVE_CHILD_ADDRESS | RAW(4 \| 8) | Address of the contributing cursor that last updated these statistics |
| PLAN_HASH_VALUE | NUMBER | Numeric representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). |
| PARSE_CALLS | NUMBER | Number of parse calls for all cursors with this SQL text and plan |
| DISK_READS | NUMBER | Number of disk reads for all cursors with this SQL text and plan |
| DIRECT_WRITES | NUMBER | Number of direct writes for all cursors with this SQL text and plan |
| BUFFER_GETS | NUMBER | Number of buffer gets for all cursors with this SQL text and plan |
| ROWS_PROCESSED | NUMBER | Total number of rows the parsed SQL statement returns |
| SERIALIZABLE_ABORTS | NUMBER | Number of times the transaction failed to serialize, producing ORA-08177 errors, per cursor |
| FETCHES | NUMBER | Number of fetches associated with the SQL statement |
| EXECUTIONS | NUMBER | Number of executions that took place on this object since it was brought into the library cache |
| END_OF_FETCH_COUNT | NUMBER | Number of times this cursor was fully executed since the cursor was brought into the library cache. The value of this statistic is not incremented when the cursor is partially executed, either because it failed during the execution or because only the first few rows produced by this cursor are fetched before the cursor is closed or re-executed. By definition, the value of the END_OF_FETCH_COUNT column should be less or equal to the value of the EXECUTIONS column. |
| LOADS | NUMBER | Number of times the object was either loaded or reloaded |
| VERSION_COUNT | NUMBER | number of cursors present in the cache with this SQL text and plan |
| INVALIDATIONS | NUMBER | Number of times this child cursor has been invalidated |
| PX_SERVERS_EXECUTIONS | NUMBER | Total number of executions performed by parallel execution servers (0 when the statement has never been executed in parallel) |
| CPU_TIME | NUMBER | CPU time (in microseconds) used by this cursor for parsing, executing, and fetching |
| ELAPSED_TIME | NUMBER | Elapsed time (in microseconds) used by this cursor for parsing, executing, and fetching. If the cursor uses parallel execution, then |

| Column | Datatype | Description |
|---|---|---|
| | | ELAPSED_TIME is the cumulative time for the query coordinator, plus all parallel query slave processes. |
| AVG_HARD_PARSE_TIME | NUMBER | Average hard parse time (in microseconds) used by this cursor |
| APPLICATION_WAIT_TIME | NUMBER | Application wait time (in microseconds) |
| CONCURRENCY_WAIT_TIME | NUMBER | Concurrency wait time (in microseconds) |
| CLUSTER_WAIT_TIME | NUMBER | Cluster wait time (in microseconds). This value is specific to Oracle RAC. It shows the total time spent waiting for all waits that are categorized under the cluster class of wait events. The value is this column is an accumulated wait time spent waiting for Oracle RAC cluster resources. |
| USER_IO_WAIT_TIME | NUMBER | User I/O wait time (in microseconds) |
| PLSQL_EXEC_TIME | NUMBER | PL/SQL execution time (in microseconds) |
| JAVA_EXEC_TIME | NUMBER | Java execution time (in microseconds) |
| SORTS | NUMBER | Number of sorts that were done for the child cursor |
| SHARABLE_MEM | NUMBER | Total shared memory (in bytes) currently occupied by all cursors with this SQL text and plan |
| TOTAL_SHARABLE_MEM | NUMBER | Total shared memory (in bytes) occupied by all cursors with this SQL text and plan if they were to be fully loaded in the shared pool (that is, cursor size) |
| TYPECHECK_MEM | NUMBER | Typecheck memory |
| IO_CELL_OFFLOAD_ELIGIBLE_BYTES | NUMBER | Number of I/O bytes which can be filtered by the Exadata storage system<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_INTERCONNECT_BYTES | NUMBER | Number of I/O bytes exchanged between Oracle Database and the storage system. Typically used for Cache Fusion or parallel queries. |
| PHYSICAL_READ_REQUESTS | NUMBER | Number of physical read I/O requests issued by the monitored SQL. The requests may not be disk reads. |
| PHYSICAL_READ_BYTES | NUMBER | Number of bytes read from disks by the monitored SQL |
| PHYSICAL_WRITE_REQUESTS | NUMBER | Number of physical write I/O requests issued by the monitored SQL |
| PHYSICAL_WRITE_BYTES | NUMBER | Number of bytes written to disks by the monitored SQL |
| EXACT_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to EXACT |

| Column | Datatype | Description |
|---|---|---|
| FORCE_MATCHING_SIGNATURE | NUMBER | Signature used when the CURSOR_SHARING parameter is set to FORCE |
| IO_CELL_UNCOMPRESSED_BYTES | NUMBER | Number of uncompressed bytes (that is, size after decompression) that are offloaded to the Exadata cells<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| IO_CELL_OFFLOAD_RETURNED_BYTES | NUMBER | Number of bytes that are returned by Exadata cell through the regular I/O path<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| DELTA_PARSE_CALLS | NUMBER | Number of parse calls for the cursor since the last Automatic Workload Repository (AWR) snapshot<br>**See Also:** *Oracle Database Concepts* for an introduction to AWR |
| DELTA_DISK_READS | NUMBER | Number of disk reads for the cursor since the last AWR snapshot |
| DELTA_DIRECT_WRITES | NUMBER | Number of direct writes for the cursor since the last AWR snapshot |
| DELTA_BUFFER_GETS | NUMBER | Number of buffer gets for the cursor since the last AWR snapshot |
| DELTA_ROWS_PROCESSED | NUMBER | Number of rows returned by the cursor since the last AWR snapshot |
| DELTA_FETCH_COUNT | NUMBER | Number of fetches for the cursor since the last AWR snapshot |
| DELTA_EXECUTION_COUNT | NUMBER | Number of executions for the cursor since the last AWR snapshot |
| DELTA_PX_SERVERS_EXECUTIONS | NUMBER | Number of executions performed by parallel execution servers since the last AWR snapshot |
| DELTA_END_OF_FETCH_COUNT | NUMBER | Number of times the cursor was fully executed since the last AWR snapshot |
| DELTA_CPU_TIME | NUMBER | CPU time (in microseconds) for the cursor since the last AWR snapshot |
| DELTA_ELAPSED_TIME | NUMBER | Database time (in microseconds) for the cursor since the last AWR snapshot |
| DELTA_APPLICATION_WAIT_TIME | NUMBER | Time spent by the cursor (in microseconds) in the Application wait class since the last AWR snapshot |
| DELTA_CONCURRENCY_TIME | NUMBER | Time spent by the cursor (in microseconds) in the Concurrency wait class since the last AWR snapshot |
| DELTA_CLUSTER_WAIT_TIME | NUMBER | Time spent by the cursor (in microseconds) in the Cluster wait class since the last AWR snapshot |
| DELTA_USER_IO_WAIT_TIME | NUMBER | Time spent by the cursor (in microseconds) in the User I/O wait class since the last AWR snapshot |
| DELTA_PLSQL_EXEC_TIME | NUMBER | Time spent by the cursor (in microseconds) executing PL/SQL since the last AWR snapshot |

| Column | Datatype | Description |
|---|---|---|
| DELTA_JAVA_EXEC_TIME | NUMBER | Time spent by the cursor (in microseconds) executing Java since the last AWR snapshot |
| DELTA_SORTS | NUMBER | Number of sorts for the cursor since the last AWR snapshot |
| DELTA_LOADS | NUMBER | Number of times the cursor was loaded since the last AWR snapshot |
| DELTA_INVALIDATIONS | NUMBER | Number of times the cursor was invalidated since the last AWR snapshot |
| DELTA_PHYSICAL_READ_REQUESTS | NUMBER | Number of physical read I/O requests for the cursor since the last AWR snapshot |
| DELTA_PHYSICAL_READ_BYTES | NUMBER | Number of bytes read from disk for the cursor since the last AWR snapshot |
| DELTA_PHYSICAL_WRITE_REQUESTS | NUMBER | Number of physical write I/O requests for the cursor since the last AWR snapshot |
| DELTA_PHYSICAL_WRITE_BYTES | NUMBER | Number of bytes written to disk for the cursor since the last AWR snapshot |
| DELTA_IO_INTERCONNECT_BYTES | NUMBER | Number of I/O bytes exchanged between the Oracle database and the storage system for the cursor since the last AWR snapshot |
| DELTA_CELL_OFFLOAD_ELIG_BYTES | NUMBER | Number of I/O bytes which can be filtered by the Exadata storage system for the cursor since the last AWR snapshot<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| DELTA_CELL_UNCOMPRESSED_BYTES | NUMBER | Number of uncompressed bytes that are offloaded to the Exadata cell for the cursor since the last AWR snapshot<br>**See Also:** Oracle Exadata Storage Server Software documentation for more information |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CON_DBID | NUMBER | The database ID of the PDB |
| OBSOLETE_COUNT | NUMBER | Number of times that a parent cursor became obsolete |

**See Also:**

- "V$SQL"
- "V$SQLAREA"

# V$SQLSTATS_PLAN_HASH

V$SQLSTATS_PLAN_HASH displays basic performance statistics for SQL cursors and contains one row per execution plan of a SQL statement (that is, one row per unique combination of SQL_ID and PLAN_HASH_VALUE).

The columns for V$SQLSTATS_PLAN_HASH are the same as those for V$SQLSTATS.

**See Also:**

"V$SQLSTATS"

# V$SQLTEXT

V$SQLTEXT displays the text of SQL statements belonging to shared SQL cursors in the SGA.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Used with HASH_VALUE to uniquely identify a cached cursor |
| HASH_VALUE | NUMBER | Used with ADDRESS to uniquely identify a cached cursor |
| SQL_ID | VARCHAR2(13) | SQL identifier of a cached cursor |
| COMMAND_TYPE | NUMBER | Code for the type of SQL statement (SELECT, INSERT, and so on) |
| PIECE | NUMBER | Number used to order the pieces of SQL text |
| SQL_TEXT | VARCHAR2(64) | A column containing one piece of the SQL text |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SQLTEXT_WITH_NEWLINES

V$SQLTEXT_WITH_NEWLINES is identical to the V$SQLTEXT view except that, to improve legibility, V$SQLTEXT_WITH_NEWLINES does not replace newlines and tabs in the SQL statement with spaces.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Used with HASH_VALUE to uniquely identify a cached cursor |
| HASH_VALUE | NUMBER | Used with ADDRESS to uniquely identify a cached cursor |
| SQL_ID | VARCHAR2(13) | SQL identifier of a cached cursor |

| Column | Datatype | Description |
|---|---|---|
| COMMAND_TYPE | NUMBER | Code for the type of SQL statement (SELECT, INSERT, and so on) |
| PIECE | NUMBER | Number used to order the pieces of SQL text |
| SQL_TEXT | VARCHAR2(64) | A column containing one piece of the SQL text |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SQLTEXT"

# V$STANDBY_EVENT_HISTOGRAM

V$STANDBY_EVENT_HISTOGRAM displays the histogram of apply lag on the physical standby. Each distinct value of apply lag has its own bucket and the count in the corresponding bucket represents the number of occurrences so far. The physical standby samples the apply lag every second and increments the corresponding bucket in the histogram.

| Column | Datatype | Description |
|---|---|---|
| NAME | VARCHAR2(64) | Name of the event (currently APPLY LAG is the only valid value) |
| TIME | NUMBER | Time duration that the bucket represents |
| UNIT | VARCHAR2(16) | Time unit (seconds, minutes, hours, or days) |
| COUNT | NUMBER | Each row is a histogram bucket for apply lag. COUNT is the number of occurrences the apply lag falls into the histogram bucket. |
| LAST_TIME_UPDATED | VARCHAR2(20) | Last time the bucket was updated by an event falling into that time duration |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$STANDBY_LOG

V$STANDBY_LOG displays information about standby redo logs. Standby redo logs are similar to online redo logs, but standby redo logs are only used on a standby database that is receiving redo data from the primary database.

| Column | Datatype | Description |
|---|---|---|
| GROUP# | NUMBER | Log group number |
| DBID | VARCHAR2(40) | Database ID of the primary database to which the standby redo logfile is assigned. If the standby redo logfile is unassigned, the value UNASSIGNED will be displayed. |
| THREAD# | NUMBER | Log thread number |
| SEQUENCE# | NUMBER | Log sequence number |
| BYTES | NUMBER | Size of the log (in bytes) |
| BLOCKSIZE | NUMBER | Block size of the logfile (512 or 4096) |
| USED | NUMBER | Number of bytes used in the log |
| ARCHIVED | VARCHAR2(3) | Archive status (YES) or (NO). See the STATUS column for further details. |
| STATUS | VARCHAR2(10) | Log status:<br>• UNASSIGNED - If ARCHIVED is NO, then the standby redo log has been archived and is again available. If ARCHIVED is YES, then the standby redo log has never been used and is available.<br>• ACTIVE - If ARCHIVED is NO, then the standby redo log is complete and waiting to be archived. If ARCHIVED is YES, then the standby redo log is currently being written to and is therefore not ready to be archived. For a given thread, there should be only one such log. |
| FIRST_CHANGE# | NUMBER | Lowest SCN in the log |
| FIRST_TIME | DATE | Time of the first SCN in the log |
| NEXT_CHANGE# | NUMBER | All redo records contained within this log will have an SCN lower than NEXT_CHANGE#. Only filled in once the log is complete. Also the lowest SCN of any redo record in the next log. |
| NEXT_TIME | DATE | All redo records contained within this log will have a timestamp lower than NEXT_TIME. Only filled in once the log is complete. Also the lowest timestamp of any redo record in the next log. |
| LAST_CHANGE# | NUMBER | All redo records contained within this log will have an SCN lower than LAST_CHANGE#. Only filled in once the log is complete. Also the lowest SCN of any redo record in the next log.<br>LAST_CHANGE# is deprecated. Use NEXT_CHANGE# instead. |
| LAST_TIME | DATE | All redo records contained within this log will have a timestamp lower than LAST_TIME. Only filled in once the log is complete. Also the lowest timestamp of any redo record in the next log.<br>LAST_TIME is deprecated. Use NEXT_TIME instead. |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$STATISTICS_LEVEL

V$STATISTICS_LEVEL displays the status of the statistics/advisories controlled by STATISTICS_LEVEL.

| Column | Datatype | Description |
|---|---|---|
| STATISTICS_NAME | VARCHAR2(64) | Name of the statistic/advisory |
| DESCRIPTION | VARCHAR2(4000) | Description of the statistic/advisory |
| SESSION_STATUS | VARCHAR2(8) | Status of the statistic/advisory for the session:<br>• ENABLED<br>• DISABLED |
| SYSTEM_STATUS | VARCHAR2(8) | System-wide status of the statistic/advisory:<br>• ENABLED<br>• DISABLED |
| ACTIVATION_LEVEL | VARCHAR2(7) | Indicates the level of STATISTICS_LEVEL that enables the statistic/advisory:<br>• BASIC<br>• TYPICAL<br>• ALL |
| STATISTICS_VIEW_NAME | VARCHAR2(64) | If there is a single view externalizing the statistic/advisory, then this column contains the name of that view. If there is no such view, then this column is null. If there are multiple views involved, then the DESCRIPTION column mentions the view names. |
| SESSION_SETTABLE | VARCHAR2(3) | Indicates whether the statistic/advisory can be set at the session level (YES) or not (NO) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"STATISTICS_LEVEL"

# V$STATNAME

V$STATNAME displays decoded statistic names for the statistics shown in the V$SESSTAT and V$SYSSTAT tables.

On some platforms, the NAME and CLASS columns contain additional operating system-specific statistics.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC# | NUMBER | Statistic number<br>**Note:** Statistics numbers are not guaranteed to remain constant from one release to another. Therefore, you should rely on the statistics name rather than its number in your applications. |
| NAME | VARCHAR2(64) | Statistic name. Names that appear in this column remain stable across Oracle Database releases, and they can be relied on by customer scripts. |
| CLASS | NUMBER | A number representing one or more statistics classes. The following class numbers are additive:<br>• 1 - User<br>• 2 - Redo<br>• 4 - Enqueue<br>• 8 - Cache<br>• 16 - OS<br>• 32 - Real Application Clusters<br>• 64 - SQL<br>• 128 - Debug |
| STAT_ID | NUMBER | Identifier of the statistic |
| DISPLAY_NAME | VARCHAR2(64) | A clearer and more descriptive name for the statistic that appears in the NAME column. Names that appear in the DISPLAY_NAME column can change across Oracle Database releases, therefore customer scripts should not rely on names that appear in the DISPLAY_NAME column across releases. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$SESSTAT" and "V$SYSSTAT"
- Statistics Descriptions for a description of each statistic
- Your operating system-specific Oracle documentation

# V$STREAMS_APPLY_COORDINATOR

V$STREAMS_APPLY_COORDINATOR displays information about each apply process coordinator. The coordinator for an apply process gets transactions from the apply process reader and passes them to apply servers. An apply process coordinator is a subcomponent of an apply process, outbound server, or inbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the coordinator's session |
| SERIAL# | NUMBER | Serial number of the coordinator's session |
| STATE | VARCHAR2(21) | State of the coordinator:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• APPLYING - Passing transactions to apply servers<br>• SHUTTING DOWN CLEANLY - Stopping without an error<br>• ABORTING - Stopping because of an apply error |
| APPLY# | NUMBER | Apply process number. An apply process coordinator is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| TOTAL_APPLIED | NUMBER | Total number of transactions applied by the apply process since the apply process was last started |
| TOTAL_WAIT_DEPS | NUMBER | Number of times since the apply process was last started that an apply server waited to apply a logical change record (LCR) in a transaction until another apply server applied a transaction because of a dependency between the transactions |
| TOTAL_WAIT_COMMITS | NUMBER | Number of times since the apply process was last started that an apply server waited to commit a transaction until another apply server committed a transaction to serialize commits |
| TOTAL_ADMIN | NUMBER | Number of administrative jobs issued since the apply process was last started |
| TOTAL_ASSIGNED | NUMBER | Number of transactions assigned to apply servers since the apply process was last started |
| TOTAL_RECEIVED | NUMBER | Total number of transactions received by the coordinator process since the apply process was last started |
| TOTAL_IGNORED | NUMBER | Number of transactions which were received by the coordinator but were ignored because they had been previously applied |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_ROLLBACKS | NUMBER | Number of transactions which were rolled back due to unexpected contention |
| TOTAL_ERRORS | NUMBER | Number of transactions applied by the apply process that resulted in an apply error since the apply process was last started |
| UNASSIGNED_COMPLETE_TXNS | NUMBER | Total number of complete transactions that the coordinator has not assigned to any apply servers |
| AUTO_TXN_BUFFER_SIZE | NUMBER | Current value of transaction buffer size. Transaction buffer size refers to the number of transactions that the apply reader can assemble ahead of apply servers. The apply process periodically adjusts the transaction buffer size. |
| LWM_TIME | DATE | Time when the message with the lowest message number was recorded. The creation time of the message with the lowest message number was also recorded at this time. |
| LWM_MESSAGE_NUMBER | NUMBER | Number of the message corresponding to the low watermark. That is, messages with a commit message number less than or equal to this message number have definitely been applied, but some messages with a higher commit message number also may have been applied. |
| LWM_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the message corresponding to the low watermark. For user-enqueued messages, time when the message corresponding to the low watermark was enqueued into the queue at the local database. |
| HWM_TIME | DATE | Time when the message with the highest message number was recorded. The creation time of the message with the highest message number was also recorded at this time. |
| HWM_MESSAGE_NUMBER | NUMBER | Number of the message corresponding to the high watermark. That is, no messages with a commit message number greater than this message number have been applied. |
| HWM_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the message corresponding to the high watermark. For user-enqueued messages, time when the message corresponding to the high watermark was enqueued into the queue at the local database. |
| STARTUP_TIME | DATE | Time when the apply process was last started |
| ELAPSED_SCHEDULE_TIME | NUMBER | Time elapsed (in hundredths of a second) scheduling messages since the apply process was last started |
| ELAPSED_IDLE_TIME | NUMBER | Elapsed idle time |
| LWM_POSITION | RAW(64) | Position of the low-watermark LCR |
| HWM_POSITION | RAW(64) | Position of the high-watermark LCR |
| PROCESSED_MESSAGE_NUMBER | NUMBER | Message number currently processed by the apply coordinator |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| ACTIVE_SERVER_COUNT[1] | NUMBER | Active server count |

[1] This column is available starting with Ora cle Database 12*c* Release 1 (12.1.0.2).

**Note:**

The ELAPSED_SCHEDULE_TIME column is only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$STREAMS_APPLY_READER

V$STREAMS_APPLY_READER displays information about each apply reader. The apply reader is a process which reads (dequeues) messages from the queue, computes message dependencies, and builds transactions. It passes the transactions on to the coordinator in commit order for assignment to the apply servers. An apply reader is a subcomponent of an apply process, outbound server, or inbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the reader's session |
| SERIAL# | NUMBER | Serial number of the reader's session |
| APPLY# | NUMBER | Apply process number. An apply process is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| STATE | VARCHAR2(36) | State of the reader:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• DEQUEUE MESSAGES - Dequeuing messages from the queue<br>• SCHEDULE MESSAGES - Computing dependencies between messages and assembling messages into transactions<br>• SPILLING - Spilling unapplied messages from memory to hard disk |

| Column | Datatype | Description |
|---|---|---|
| | | • PAUSED - WAITING FOR DDL TO COMPLETE - Waiting for a data definition language (DDL) logical change record (LCR) to be applied |
| TOTAL_MESSAGES_DEQUEUED | NUMBER | Total number of messages dequeued since the apply process was last started |
| TOTAL_MESSAGES_SPILLED | NUMBER | Number of messages spilled by the reader since the apply process was last started |
| DEQUEUE_TIME | DATE | Time when the last message was received |
| DEQUEUED_MESSAGE_NUMBER | NUMBER | Number of the last message received |
| DEQUEUED_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the last message received. For user-enqueued messages, time when the message was enqueued into the queue at the local database. |
| SGA_USED | NUMBER | Amount (in bytes) of SGA memory used by the apply process since it was last started |
| ELAPSED_DEQUEUE_TIME | NUMBER | Time elapsed (in hundredths of a second) dequeuing messages since the apply process was last started |
| ELAPSED_SCHEDULE_TIME | NUMBER | Time elapsed (in hundredths of a second) scheduling messages since the apply process was last started. Scheduling includes computing dependencies between messages and assembling messages into transactions. |
| ELAPSED_SPILL_TIME | NUMBER | Elapsed time (in hundredths of a second) spent spilling messages since the apply process was last started |
| LAST_BROWSE_NUM | NUMBER | Reserved for internal use |
| OLDEST_SCN_NUM | NUMBER | Oldest SCN |
| LAST_BROWSE_SEQ | NUMBER | Reserved for internal use |
| LAST_DEQ_SEQ | NUMBER | Last dequeue sequence number |
| OLDEST_XIDUSN | NUMBER | Transaction ID undo segment number of the oldest transaction that either has been applied or is being applied |
| OLDEST_XIDSLT | NUMBER | Transaction ID slot number of the oldest transaction that either has been applied or is being applied |
| OLDEST_XIDSQN | NUMBER | Transaction ID sequence number of the oldest transaction that either has been applied or is being applied |
| SPILL_LWM_SCN | NUMBER | Spill low-watermark SCN |
| PROXY_SID | NUMBER | When the apply process uses combined capture and apply, the session ID of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is 0. |
| PROXY_SERIAL | NUMBER | When the apply process uses combined capture and apply, the serial number of the propagation receiver that is responsible for |

| Column | Datatype | Description |
|---|---|---|
| | | direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is `0`. |
| PROXY_SPID | VARCHAR2(12) | When the apply process uses combined capture and apply, the process identification number of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is `0`. |
| CAPTURE_BYTES_RECEIVED | NUMBER | When the apply process uses combined capture and apply, the number of bytes received by the apply process from the capture process since the apply process last started. If the apply process does not use combined capture and apply, then this column is not populated. |
| DEQUEUED_POSITION | RAW(64) | Dequeued position. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| LAST_BROWSE_POSITION | RAW(64) | Reserved for internal use |
| OLDEST_POSITION | RAW(64) | The earliest position of the transactions currently being dequeued and applied. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| SPILL_LWM_POSITION | RAW(64) | Spill low-watermark position. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| OLDEST_TRANSACTION_ID | VARCHAR2(128) | Oldest transaction ID |
| TOTAL_LCRS_WITH_DEP | NUMBER | Total number of LCRs with row-level dependencies since the apply process last started |
| TOTAL_LCRS_WITH_WMDEP | NUMBER | Total number of LCRs with watermark dependencies since the apply process last started.<br>A watermark dependency occurs when an apply process must wait until the apply process's low watermark reaches a particular threshold. |
| TOTAL_IN_MEMORY_LCRS | NUMBER | Total number of LCRs currently in memory |
| SGA_ALLOCATED | NUMBER | The total amount of shared memory (in bytes) allocated from the Streams pool for the apply process since the apply process last started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The ELAPSED_DEQUEUE_TIME and ELAPSED_SCHEDULE_TIME columns are only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$STREAMS_APPLY_SERVER

V$STREAMS_APPLY_SERVER displays information about each apply server and its activities. An apply server receives messages from the apply coordinator for an apply process. For each message received, an apply server either applies the message or sends the message to the appropriate apply handler. An apply server is a subcomponent of an apply process, outbound server, or inbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the apply server's session |
| SERIAL# | NUMBER | Serial number of the apply server's session |
| APPLY# | NUMBER | Apply process number. An apply process is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| SERVER_ID | NUMBER | Parallel execution server number of the apply server |
| STATE | VARCHAR2(20) | State of the apply server:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• RECORD LOW-WATERMARK - Performing an administrative job that maintains information about the apply progress, which is used in the ALL_APPLY_PROGRESS and DBA_APPLY_PROGRESS data dictionary views<br>• ADD PARTITION - Performing an administrative job that adds a partition that is used for recording information about in-progress transactions<br>• DROP PARTITION - Performing an administrative job that purges rows that were used to record information about in-progress transactions<br>• EXECUTE TRANSACTION - Applying a transaction<br>• WAIT COMMIT - Waiting to commit a transaction until all other transactions with a lower commit SCN are applied. This state is possible only if the COMMIT_SERIALIZATION apply process parameter is set to a value other than DEPENDENT_TRANSACTIONS and the PARALLELISM apply process parameter is set to a value greater than 1.<br>• WAIT DEPENDENCY - Waiting to apply a logical change record (LCR) in a transaction until another transaction, on |

| Column | Datatype | Description |
|---|---|---|
| | | which it has a dependency, is applied. This state is possible only if the PARALLELISM apply process parameter is set to a value greater than 1.<br>• ROLLBACK TRANSACTION - Rolling back a transaction<br>• TRANSACTION CLEANUP - Cleaning up an applied transaction, which includes removing LCRs from the apply process's queue<br>• WAIT FOR CLIENT - Waiting for an XStream client application to request more LCRs<br>• WAIT FOR NEXT CHUNK - Waiting for the next set of LCRs for a large transaction |
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction currently being applied |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction currently being applied |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction currently being applied |
| COMMITSCN | NUMBER | Commit system change number (SCN) of the transaction currently being applied |
| DEP_XIDUSN | NUMBER | Transaction ID undo segment number of a transaction on which the transaction being applied by this apply server depends |
| DEP_XIDSLT | NUMBER | Transaction ID slot number of a transaction on which the transaction being applied by this apply server depends |
| DEP_XIDSQN | NUMBER | Transaction ID sequence number of a transaction on which the transaction being applied by this apply server depends |
| DEP_COMMITSCN | NUMBER | Commit system change number (SCN) of the transaction on which this apply server depends |
| MESSAGE_SEQUENCE | NUMBER | Number of the current message being applied by the apply server. This value is reset to 1 at the beginning of each transaction. |
| TOTAL_ASSIGNED | NUMBER | Total number of transactions assigned to the apply server since the apply process was last started |
| TOTAL_ADMIN | NUMBER | Total number of administrative jobs done by the apply server since the apply process was last started. See the STATE information in this view for the types of administrative jobs. |
| TOTAL_ROLLBACKS | NUMBER | Number of transactions assigned to this server which were rolled back |
| TOTAL_MESSAGES_APPLIED | NUMBER | Total number of messages applied by this apply server since the apply process was last started |
| APPLY_TIME | DATE | Time the last message was applied |
| APPLIED_MESSAGE_NUMBER | NUMBER | Number of the last message applied |

| Column | Datatype | Description |
|---|---|---|
| APPLIED_MESSAGE_CREATE_TIME | DATE | Creation time at the source database of the last captured message applied. No information about user-enqueued messages is recorded in this column. |
| ELAPSED_DEQUEUE_TIME | NUMBER | Time elapsed (in hundredths of a second) dequeuing messages since the apply process was last started |
| ELAPSED_APPLY_TIME | NUMBER | Time elapsed (in hundredths of a second) applying messages since the apply process was last started |
| COMMIT_POSITION | RAW(64) | Commit position of the transaction. This column is populated only for an apply process that is functioning as an XStream outbound server or inbound server. |
| DEP_COMMIT_POSITION | RAW(64) | Commit position of the transaction the slave depends on. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| LAST_APPLY_POSITION | RAW(64) | For inbound servers, the position of the last message applied; for outbound servers, the position of the last message sent to the XStream client application. This column is populated only for an apply process that is functioning as an XStream outbound server or inbound server. |
| TRANSACTION_ID | VARCHAR2(128) | Transaction ID that the slave is applying. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| DEP_TRANSACTION_ID | VARCHAR2(128) | Transaction ID of the transaction the slave depends on. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The ELAPSED_DEQUEUE_TIME and ELAPSED_APPLY_TIME columns are only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$STREAMS_CAPTURE

V$STREAMS_CAPTURE displays information about each capture process that sends LCRs to an XStream outbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier of the capture process |
| SERIAL# | NUMBER | Session serial number of the capture process session |
| CAPTURE# | NUMBER | Capture process number. A capture process is an Oracle background process, prefixed by cp. |
| CAPTURE_NAME | VARCHAR2(30) | Name of the capture process |
| LOGMINER_ID | NUMBER | Session ID of the Oracle LogMiner session associated with the capture process |
| STARTUP_TIME | DATE | Time when the capture process was last started |
| STATE | VARCHAR2(551) | State of the capture process:<br>• ABORTING - Aborting.<br>• CAPTURING CHANGES - Scanning the redo log for changes that satisfy the capture process rule sets.<br>• CREATING LCR - Converting a change into an LCR.<br>• DICTIONARY INITIALIZATION - Processing a dictionary build.<br>• ENQUEUING MESSAGE - Enqueuing an LCR that satisfies the capture process rule sets into the capture process queue.<br>• EVAULATING RULE - Evaluating a change against a capture process rule set.<br>• INITIALIZING - Starting up.<br>• LOADING (step *X* of *Y*) - Processing information from a dictionary build and currently at step *X* in a process that involves *Y* steps, where *X* and *Y* are numbers.<br>• MINING (PROCESSED SCN = *scn_value*) - Mining a dictionary build at the SCN *scn_value*.<br>• PAUSED FOR FLOW CONTROL - Unable to enqueue LCRs either because of low memory or because propagations and outbound servers are consuming messages slower than the capture process is creating them. This state indicates flow control that is used to reduce spilling of captured LCRs when propagation or apply has fallen behind or is unavailable.<br>• SHUTTING DOWN - Stopping.<br>• SUSPENDED FOR AUTO SPLIT/MERGE - Waiting for a merge operation to complete.<br>• WAITING FOR DICTIONARY REDO - Waiting for redo log files containing the dictionary build related to the first SCN to be added to the capture process session. A capture process cannot begin to scan the redo log files until all of the log files containing the dictionary build have been added.<br>• WAITING FOR INACTIVE DEQUEUERS - Waiting for the capture process's queue subscribers to start. The capture process stops enqueuing LCRs if there are no active subscribers to the queue.. |

| Column | Datatype | Description |
|---|---|---|
| | | • WAITNG FOR *n* SUBSCRIBER(S) INITIALIZING - Waiting for outbound servers that receive LCRs from the capture process to start, where n is the number of outbound servers.<br>• WAITING FOR REDO - Waiting for new redo log files to be added to the capture process session. The capture process has finished processing all of the redo log files added to its session. This state is possible if there is no activity at a source database. For a downstream capture process, this state is possible if the capture process is waiting for new log files to be added to its session.<br>• WAITING FOR THE BUFFERED QUEUE TO SHRINK - Waiting for the buffered queue to change to a smaller size. The buffered queue shrinks when there is a memory limitation or when an administrator reduces its size.<br>• WAITING FOR TRANSACTION - Waiting for Oracle LogMiner to provide more transactions. |
| TOTAL_PREFILTER_DISCARDED | NUMBER | Total number of prefiltered messages discarded |
| TOTAL_PREFILTER_KEPT | NUMBER | Total number of prefiltered messages kept |
| TOTAL_PREFILTER_EVALUATIONS | NUMBER | Total number of prefilter evaluations |
| TOTAL_MESSAGES_CAPTURED | NUMBER | Total number of redo entries passed by LogMiner to the capture process for detailed rule evaluation since the capture process last started. A capture process converts a redo entry into a message and performs detailed rule evaluation on the message when capture process prefiltering cannot discard the change. |
| CAPTURE_TIME | DATE | Time when the most recent message was captured |
| CAPTURE_MESSAGE_NUMBER | NUMBER | Number of the most recently captured message |
| CAPTURE_MESSAGE_CREATE_TIME | DATE | Creation time of the most recently captured message |
| TOTAL_MESSAGES_CREATED | NUMBER | Count associated with ELAPSED_LCR_TIME to calculate rate |
| TOTAL_FULL_EVALUATIONS | NUMBER | Count associated with ELAPSED_RULE_TIME to calculate rate |
| TOTAL_MESSAGES_ENQUEUED | NUMBER | Total number of messages enqueued since the capture process was last started |
| ENQUEUE_TIME | DATE | Time when the last message was enqueued |
| ENQUEUE_MESSAGE_NUMBER | NUMBER | Number of the last enqueued message |
| ENQUEUE_MESSAGE_CREATE_TIME | DATE | Creation time of the last enqueued message |

| Column | Datatype | Description |
|---|---|---|
| AVAILABLE_MESSAGE_NUMBER | NUMBER | For local capture, the last redo SCN flushed to the logfiles. For downstream capture, the last SCN added to LogMiner via the archive logs. |
| AVAILABLE_MESSAGE_CREATE_TIME | DATE | For local capture, the time the SCN was written to the logfile. For downstream capture, the time the most recent archive log (containing the most recent SCN) was added to LogMiner. |
| ELAPSED_CAPTURE_TIME | NUMBER | Elapsed time (in hundredths of a second) scanning for changes in the redo log since the capture process was last started |
| ELAPSED_RULE_TIME | NUMBER | Elapsed time (in hundredths of a second) evaluating rules since the capture process was last started |
| ELAPSED_ENQUEUE_TIME | NUMBER | Elapsed time (in hundredths of a second) enqueuing messages since the capture process was last started |
| ELAPSED_LCR_TIME | NUMBER | Elapsed time (in hundredths of a second) creating logical change records (LCRs) since the capture process was last started |
| ELAPSED_REDO_WAIT_TIME | NUMBER | Elapsed time (in hundredths of a second) spent by the capture process in the `WAITING FOR REDO` state |
| ELAPSED_PAUSE_TIME | NUMBER | Elapsed flow control pause time (in hundredths of a second) |
| STATE_CHANGED_TIME | DATE | Time at which the state of the capture process changed |
| APPLY_NAME | VARCHAR2(30) | Reserved for internal use |
| APPLY_DBLINK | VARCHAR2(128) | Reserved for internal use |
| APPLY_MESSAGES_SENT | NUMBER | Reserved for internal use |
| APPLY_BYTES_SENT | NUMBER | Reserved for internal use |
| OPTIMIZATION | NUMBER | Indicates whether a capture process uses combined capture and apply (greater than zero) or a capture process does not use combined capture and apply (`0`) |
| PURPOSE | CHAR(7) | Purpose of the capture process:<br>• `Streams` - The capture process is part of an Oracle Streams configuration |
| SGA_USED | NUMBER | The total amount of shared memory (in bytes) currently used by the capture process out of the amount allocated (`SGA_ALLOCATED`) |
| SGA_ALLOCATED | NUMBER | The total amount of shared memory (in bytes) allocated from the Streams pool for the capture process |
| BYTES_OF_REDO_MINED | VARCHAR2(64) | The total amount of redo data mined (in bytes) since the capture process last started |
| SESSION_RESTART_SCN | VARCHAR2(64) | The SCN from which the capture process started mining redo data when it was last started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The ELAPSED_CAPTURE_TIME, ELAPSED_RULE_TIME, ELAPSED_ENQUEUE_TIME, ELAPSED_LCR_TIME, and ELAPSED_REDO_WAIT_TIME columns are only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$STREAMS_MESSAGE_TRACKING

V$STREAMS_MESSAGE_TRACKING displays information about LCRs tracked through the stream that are processed by each Streams client. Use the DBMS_STREAMS_ADM.SET_MESSAGE_TRACKING procedure to specify a tracking label that becomes part of each LCR generated by the current session.

When the actions parameter in the DBMS_STREAMS_ADM.SET_MESSAGE_TRACKING procedure is set to DBMS_STREAMS_ADM.ACTION_MEMORY, information about the LCRs is tracked in memory, and this view is populated with information about the LCRs. Currently, DBMS_STREAMS_ADM.ACTION_MEMORY is the only valid setting for the actions parameter in the procedure.

| Column | Datatype | Description |
|---|---|---|
| TRACKING_LABEL | VARCHAR2(30) | User-specified tracking label |
| TAG | RAW(30) | First 30 bytes of the tag of the LCR |
| COMPONENT_NAME | VARCHAR2(30) | Name of the component that processed the LCR |
| COMPONENT_TYPE | VARCHAR2(30) | Type of the component that processed the LCR |
| ACTION | VARCHAR2(50) | Action performed on the LCR |
| ACTION_DETAILS | VARCHAR2(100) | Details of the action |
| TIMESTAMP | TIMESTAMP(9) WITH TIME ZONE | Time when the action was performed |

| Column | Datatype | Description |
|---|---|---|
| MESSAGE_CREATION_TIME | DATE | Time when the message was created |
| MESSAGE_NUMBER | NUMBER | SCN of the message |
| TRACKING_ID | RAW(16) | Globally unique OID of the LCR |
| SOURCE_DATABASE_NAME | VARCHAR2(128) | Name of the source database |
| OBJECT_OWNER | VARCHAR2(30) | Owner of the object |
| OBJECT_NAME | VARCHAR2(30) | Name of the object |
| XID | VARCHAR2(128) | Transaction ID |
| COMMAND_TYPE | VARCHAR2(30) | Command type of the LCR |
| MESSAGE_POSITION | RAW(64) | Reserved for internal use |
| PURPOSE | CHAR(7) | Purpose of the LCR stream:<br>• Streams - The LCR stream flows through an Oracle Streams configuration |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_STREAMS_ADM.SET_MESSAGE_TRACKING procedure

# V$STREAMS_POOL_ADVICE

V$STREAMS_POOL_ADVICE displays information about the estimated count of spilled or unspilled messages and the associated time spent in the spill or unspill activity for different Streams pool sizes. The sizes range from 10% to 200% of the current Streams pool size, in equal intervals. The value of the interval depends on the current size of the Streams pool.

| Column | Datatype | Description |
|---|---|---|
| STREAMS_POOL_SIZE_FOR_ESTIMATE | NUMBER | Streams pool size (in megabytes) for the estimate. The size ranges from values smaller than the current Streams pool size to values larger than the current Streams pool size, and there is a separate row for each increment. There is always an entry that shows the current Streams pool size, and there are always 20 increments. The range and the size of the increments depend on the current size of the Streams pool. |

| Column | Datatype | Description |
|---|---|---|
| STREAMS_POOL_SIZE_FACTOR | NUMBER | Size factor with respect to the current Streams pool size |
| ESTD_SPILL_COUNT | NUMBER | Estimated count of messages spilled from the Streams pool |
| ESTD_SPILL_TIME | NUMBER | Estimated elapsed time (in seconds) to spill |
| ESTD_UNSPILL_COUNT | NUMBER | Estimated count of unspills (read back from disk) |
| ESTD_UNSPILL_TIME | NUMBER | Estimated elapsed time (in seconds) to unspill |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Streams Concepts and Administration* for more information about this view

# V$STREAMS_POOL_STATISTICS

V$STREAMS_POOL_STATISTICS displays information about the current Streams pool usage percentage.

| Column | Datatype | Description |
|---|---|---|
| TOTAL_MEMORY_ALLOCATED | NUMBER | Total memory allocated to the Streams pool (in bytes). It should always be less than the current size of the Streams pool. You can get the percentage of the Streams pool used by dividing TOTAL_MEMORY_ALLOCATED by CURRENT_SIZE. |
| CURRENT_SIZE | NUMBER | Current size of the Streams pool (in bytes) |
| SGA_TARGET_VALUE | NUMBER | Value of SGA_TARGET. Used to determine whether or not streams pool automatic tuning is enabled. This should be set even if MEMORY_TARGET is set and SGA_TARGET is not set. |
| SHRINK_PHASE | NUMBER | This only pertains to the Streams pool in an automatic tuning environment (SGA_TARGET and MEMORY_TARGET set). In this case, this shows whether or not the Streams pool is being asked to shrink. During the shrink phase, enqueues are blocked, flow control is enabled for all components, and cached memory is returned to the SGA. |
| ADVICE_DISABLED | NUMBER | This determines whether or not Streams pool advice in V$STREAMS_POOL_ADVICE as well as all statistics gathering related to auto-tuning the Streams pool have been disabled. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$STREAMS_TRANSACTION

V$STREAMS_TRANSACTION displays information about transactions that are being processed by capture processes or apply processes. This view can be used to identify long running transactions and to determine how many logical change records (LCRs) are being processed in each transaction. This view only contains information about captured LCRs. It does not contain information about user-enqueued LCRs or user messages.

This view only shows information about LCRs that are being processed because they satisfied the rule sets for the Streams process at the time of the query. For capture processes, this view only shows information about changes in transactions that the capture process has converted to LCRs. It does not show information about all the active transactions present in the redo log. For apply processes, this view only shows information about LCRs that the apply process has dequeued. It does not show information about LCRs in the apply process's queue.

Information about a transaction remains in the view until the transaction commits or until the entire transaction is rolled back.

| Column | Datatype | Description |
|---|---|---|
| STREAMS_NAME | VARCHAR2(30) | Name of the Streams process |
| STREAMS_TYPE | VARCHAR2(20) | Type of the Streams process:<br>• CAPTURE<br>• APPLY<br>• PROPAGATION_SENDER |
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction |
| CUMULATIVE_MESSAGE_COUNT | NUMBER | Number of LCRs processed in the transaction. If the Streams process is restarted while the transaction is being processed, then this column shows the number of LCRs processed in the transaction since the Streams process was started. |
| TOTAL_MESSAGE_COUNT | NUMBER | Total Number of LCRs processed in the transaction by an apply process. This column does not pertain to capture processes. |
| FIRST_MESSAGE_TIME | DATE | Timestamp of the first LCR processed in the transaction. If a capture process is restarted while the transaction is being processed, then this column shows the timestamp of the first LCR processed after the capture process was started. |
| FIRST_MESSAGE_NUMBER | NUMBER | System change number (SCN) of the first message in the transaction. If a capture process is restarted while the transaction |

| Column | Datatype | Description |
|---|---|---|
| | | is being processed, then this column shows the SCN of the first message processed after the capture process was started. |
| LAST_MESSAGE_TIME | DATE | Timestamp of the last LCR processed in the transaction |
| LAST_MESSAGE_NUMBER | NUMBER | SCN of the most recent message encountered for the transaction |
| FIRST_MESSAGE_POSITION | RAW(64) | Reserved for internal use |
| LAST_MESSAGE_POSITION | RAW(64) | Reserved for internal use |
| TRANSACTION_ID | VARCHAR2(128) | Transaction ID |
| PURPOSE | CHAR(7) | Purpose of the components that are processing the transaction:<br>• `Streams` - The transaction is being processed in an Oracle Streams configuration |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SUBCACHE

V$SUBCACHE displays information about the subordinate caches currently loaded into library cache memory. The view walks through the library cache, printing out a row for each loaded subordinate cache per library cache object.

| Column | Datatype | Description |
|---|---|---|
| OWNER_NAME | VARCHAR2(64) | Owner of the object containing these cache entries |
| NAME | VARCHAR2(1000) | Object Name |
| TYPE | NUMBER | Object Type |
| HEAP_NUM | NUMBER | Heap number containing this subordinate cache |
| CACHE_ID | NUMBER | Subordinate cache ID |
| CACHE_CNT | NUMBER | Number of entries for this cache in this object |
| HEAP_SZ | NUMBER | Amount of extent space allocated to this heap |
| HEAP_ALOC | NUMBER | Amount of extent space allocated from this heap |
| HEAP_USED | NUMBER | Amount of space utilized in this heap |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SUBSCR_REGISTRATION_STATS

V$SUBSCR_REGISTRATION_STATS displays information for diagnosability of notifications.

| Column | Datatype | Description |
|---|---|---|
| REG_ID | NUMBER | Registration identifier |
| NUM_NTFNS | NUMBER | Number of notifications |
| NUM_GROUPING_NTFNS | NUMBER | Number of grouping notifications |
| NUM_NTFNS_CURRENT_GROUP | NUMBER | Number of events received in the current group |
| LAST_NTFN_START_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the last notification was started |
| LAST_NTFN_SENT_TIME | TIMESTAMP(3) WITH TIME ZONE | Time when the last notification was sent |
| TOTAL_EMON_LATENCY | NUMBER | Total EMON latency (time taken by the EMON slave to process notifications) |
| EMON# | NUMBER | Active EMON slave serving the registration |
| ALL_EMON_SERVERS | RAW(2000) | EMON slaves that served the registration |
| TOTAL_PAYLOAD_BYTES_SENT | NUMBER | Total payload bytes sent |
| NUM_RETRIES | NUMBER | Number of retries in sending notifications |
| TOTAL_PLSQL_EXEC_TIME | NUMBER | Total PL/SQL callback execution time (relevant only for PL/SQL notifications) |
| LAST_ERR | VARCHAR2(90) | Last error message |
| LAST_ERR_TIME | TIMESTAMP(3) WITH TIME ZONE | Time of the last error |
| LAST_UPDATE_TIME | TIMESTAMP(3) WITH TIME ZONE | Time of the last update |
| NUM_PENDING_NTFNS | NUMBER | Number of notifications pending to be sent |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_PENDING_NTFN_BYTES | NUMBER | Total number of bytes for notifications pending to be sent |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYS_OPTIMIZER_ENV

V$SYS_OPTIMIZER_ENV displays the contents of the optimizer environment for the instance. The optimizer environment stores the value of the main parameters used by the Oracle optimizer when building the execution plan of a SQL statement. Hence, modifying the value of one or more of these parameters (for example, by issuing an ALTER SYSTEM statement) could lead to plan changes.

The parameters displayed by this view are either regular initialization parameters (such as OPTIMIZER_FEATURES_ENABLE) or pseudo parameters (such as ACTIVE_INSTANCE_COUNT).

| Column | Datatype | Description |
|---|---|---|
| ID | NUMBER | Unique identifier of the parameter in the optimizer environment |
| NAME | VARCHAR2(40) | Name of the parameter |
| SQL_FEATURE | VARCHAR2(64) | Associated feature control ID |
| ISDEFAULT | VARCHAR2(3) | Indicates whether the parameter is set to the default value (YES) or not (NO) |
| VALUE | VARCHAR2(25) | Value of the parameter |
| DEFAULT_VALUE | VARCHAR2(25) | Default value of the parameter |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "OPTIMIZER_FEATURES_ENABLE"
- "ACTIVE_INSTANCE_COUNT"

# V$SYS_TIME_MODEL

V$SYS_TIME_MODEL displays the system-wide accumulated times for various operations. The time reported is the total elapsed or CPU time (in microseconds). Any timed operation will buffer at most 5 seconds of time data. Specifically, this means that if a timed operation (such as SQL execution) takes a long period of time to perform, the data published to this view is at most missing 5 seconds of the time accumulated for the operation.

The time values are 8-byte integers and can therefore hold approximately 580,000 years worth of time before wrapping. Background process time is not included in a statistic value unless the statistic is specifically for background processes.

| Column | Datatype | Description |
|---|---|---|
| STAT_ID | NUMBER | Statistic identifier for the time statistic |
| STAT_NAME | VARCHAR2(64) | Name of the statistic (see Table 9-1) |
| VALUE | NUMBER | Amount of time (in microseconds) that the system has spent in this operation |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view returns instance-wide data and a value of 0 in the CON_ID column when queried from the root of a CDB.

# V$SYSAUX_OCCUPANTS

V$SYSAUX_OCCUPANTS displays SYSAUX tablespace occupant information.

| Column | Datatype | Description |
|---|---|---|
| OCCUPANT_NAME | VARCHAR2(64) | Occupant name |
| OCCUPANT_DESC | VARCHAR2(64) | Occupant description |
| SCHEMA_NAME | VARCHAR2(64) | Schema name for the occupant |
| MOVE_PROCEDURE | VARCHAR2(64) | Name of the move procedure; null if not applicable |
| MOVE_PROCEDURE_DESC | VARCHAR2(64) | Description of the move procedure |
| SPACE_USAGE_KBYTES | NUMBER | Current space usage of the occupant (in KB) |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYSMETRIC

V$SYSMETRIC displays the system metric values captured for the most current time interval for both the long duration (60-second) and short duration (15-second) system metrics.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | Metric group ID |
| METRIC_ID | NUMBER | Metric ID |
| METRIC_NAME | VARCHAR2(64) | Metric name |
| VALUE | NUMBER | Metric value |
| METRIC_UNIT | VARCHAR2(64) | Metric unit description |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYSMETRIC_HISTORY

V$SYSMETRIC_HISTORY displays all system metric values available in the database. Both long duration (60-second with 1 hour history) and short duration (15-second with one-interval only) metrics are displayed by this view.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |

| Column | Datatype | Description |
|---|---|---|
| GROUP_ID | NUMBER | Metric group ID |
| METRIC_ID | NUMBER | Metric ID |
| METRIC_NAME | VARCHAR2(64) | Metric name |
| VALUE | NUMBER | Metric value |
| METRIC_UNIT | VARCHAR2(64) | Metric unit description |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYSMETRIC_SUMMARY

V$SYSMETRIC_SUMMARY displays a summary of all system Metric values for the long-duration system metrics. The average, maximum value, minimum value, and the value of one standard deviation for the last hour are displayed for each metric item.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| GROUP_ID | NUMBER | Metric group ID |
| METRIC_ID | NUMBER | Metric ID |
| METRIC_NAME | VARCHAR2(64) | Metric name |
| NUM_INTERVAL | NUMBER | Number of intervals observed |
| MAXVAL | NUMBER | Maximum value observed |
| MINVAL | NUMBER | Minimum value observed |
| AVERAGE | NUMBER | Average value over the period |
| STANDARD_DEVIATION | NUMBER | One standard deviation |
| METRIC_UNIT | VARCHAR2(64) | Metric unit description |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYSSTAT

V$SYSSTAT displays system statistics. To find the name of the statistic associated with each statistic number (STATISTIC#), query the V$STATNAME view.

| Column | Datatype | Description |
|---|---|---|
| STATISTIC# | NUMBER | Statistic number<br>**Note:** Statistics numbers are not guaranteed to remain constant from one release to another. Therefore, you should rely on the statistics name rather than its number in your applications. |
| NAME | VARCHAR2(64) | Statistic name. You can get a complete listing of statistic names by querying the V$STATNAME view. |
| CLASS | NUMBER | A number representing one or more statistics class. The following class numbers are additive:<br>• 1 - User<br>• 2 - Redo<br>• 4 - Enqueue<br>• 8 - Cache<br>• 16 - OS<br>• 32 - Real Application Clusters<br>• 64 - SQL<br>• 128 - Debug |
| VALUE | NUMBER | Statistic value |
| STAT_ID | NUMBER | Identifier of the statistic |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view returns instance-wide data and a value of 0 in the CON_ID column when queried from the root of a CDB.

**See Also:**

"V$STATNAME" and Statistics Descriptions

# V$SYSTEM_CURSOR_CACHE

V$SYSTEM_CURSOR_CACHE displays system wide information on cursor usage.

**See Also:**

"V$SESSION_CURSOR_CACHE"

| Column | Datatype | Description |
|---|---|---|
| OPENS | NUMBER | Cumulative total of cursor opens |
| HITS | NUMBER | Cumulative total of cursor open hits |
| HIT_RATIO | NUMBER | Ratio of the number of times an open cursor was found divided by the number of times a cursor was sought |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYSTEM_EVENT

V$SYSTEM_EVENT displays information on total waits for an event. Note that the TIME_WAITED and AVERAGE_WAIT columns will contain a value of zero on those platforms that do not support a fast timing mechanism. If you are running on one of these platforms and you want this column to reflect true wait times, then you must set TIMED_STATISTICS to TRUE in the parameter file; doing this will have a small negative effect on system performance.

**See Also:**

"TIMED_STATISTICS"

| Column | Datatype | Description |
|---|---|---|
| EVENT | VARCHAR2(64) | Name of the wait event |
| TOTAL_WAITS | NUMBER | Total number of waits for the event |
| TOTAL_TIMEOUTS | NUMBER | Total number of timeouts for the event |
| TIME_WAITED | NUMBER | Total amount of time waited for the event (in hundredths of a second) |
| AVERAGE_WAIT | NUMBER | Average amount of time waited for the event (in hundredths of a second) |

| Column | Datatype | Description |
|---|---|---|
| TIME_WAITED_MICRO | NUMBER | Total amount of time waited for the event (in microseconds) |
| TOTAL_WAITS_FG | NUMBER | Total number of waits for the event, from foreground sessions |
| TOTAL_TIMEOUTS_FG | NUMBER | Total number of timeouts for the event, from foreground sessions |
| TIME_WAITED_FG | NUMBER | Amount of time waited for the event (in hundredths of a second), from foreground sessions |
| AVERAGE_WAIT_FG | NUMBER | Average amount of time waited for the event (in hundredths of a second), from foreground sessions |
| TIME_WAITED_MICRO_FG | NUMBER | Amount of time waited for the event (in microseconds), from foreground sessions |
| EVENT_ID | NUMBER | Identifier of the wait event |
| WAIT_CLASS_ID | NUMBER | Identifier of the class of the wait event |
| WAIT_CLASS# | NUMBER | Number of the class of the wait event |
| WAIT_CLASS | VARCHAR2(64) | Name of the class of the wait event |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view returns instance-wide data and a value of 0 in the CON_ID column when queried from the root of a CDB.

# V$SYSTEM_FIX_CONTROL

V$SYSTEM_FIX_CONTROL displays information about Fix Control (enabled/disabled) at the system level.

| Column | Datatype | Description |
|---|---|---|
| BUGNO | NUMBER | Bug number (as fix control identifier) |
| VALUE | NUMBER | Current value set for the fix control |
| SQL_FEATURE | VARCHAR2(64) | Feature control ID |
| DESCRIPTION | VARCHAR2(64) | Description of the fix control |
| OPTIMIZER_FEATURE_ENABLE | VARCHAR2(25) | Version on (and after) which the fix is enabled by default |
| EVENT | NUMBER | Event formerly used to control this fix |

| Column | Datatype | Description |
|---|---|---|
| IS_DEFAULT | NUMBER | Indicates whether the current value is the same as the default (1) or not (0) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYSTEM_PARAMETER

V$SYSTEM_PARAMETER displays information about the initialization parameters that are currently in effect for the instance. A new session inherits parameter values from the instance-wide values.

| Column | Datatype | Description |
|---|---|---|
| NUM | NUMBER | Parameter number |
| NAME | VARCHAR2(80) | Name of the parameter |
| TYPE | NUMBER | Parameter type:<br>• 1 - Boolean<br>• 2 - String<br>• 3 - Integer<br>• 4 - Parameter file<br>• 5 - Reserved<br>• 6 - Big integer |
| VALUE | VARCHAR2(4000) | Instance-wide parameter value |
| DISPLAY_VALUE | VARCHAR2(4000) | Parameter value in a user-friendly format. For example, if the VALUE column shows the value 262144 for a big integer parameter, then the DISPLAY_VALUE column will show the value 256K. |
| DEFAULT_VALUE | VARCHAR2(255) | Default value for this parameter |
| ISDEFAULT | VARCHAR2(9) | Indicates whether the parameter is set to the default value (TRUE) or the parameter value was specified in the parameter file (FALSE) |
| ISSES_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be changed with ALTER SESSION (TRUE) or not (FALSE) |
| ISSYS_MODIFIABLE | VARCHAR2(9) | Indicates whether the parameter can be changed with ALTER SYSTEM and when the change takes effect:<br>• IMMEDIATE - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect immediately.<br>• DEFERRED - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect in subsequent sessions. |

| Column | Datatype | Description |
|---|---|---|
| | | • FALSE - Parameter cannot be changed with ALTER SYSTEM unless a server parameter file was used to start the instance. The change takes effect in subsequent instances. |
| ISPDB_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be modified inside a PDB (TRUE) or not (FALSE).<br>In a non-CDB, the value of this column is NULL. |
| ISINSTANCE_MODIFIABLE | VARCHAR2(5) | For parameters that can be changed with ALTER SYSTEM, indicates whether the value of the parameter can be different for every instance (TRUE) or whether the parameter must have the same value for all Real Application Clusters instances (FALSE). If the ISSYS_MODIFIABLE column is FALSE, then this column is always FALSE. |
| ISMODIFIED | VARCHAR2(8) | Indicates how the parameter was modified. If an ALTER SYSTEM was performed, the value will be MODIFIED. |
| ISADJUSTED | VARCHAR2(5) | Indicates whether Oracle adjusted the input value to a more suitable value (for example, the parameter value should be prime, but the user input a non-prime number, so Oracle adjusted the value to the next prime number) |
| ISDEPRECATED | VARCHAR2(5) | Indicates whether the parameter has been deprecated (TRUE) or not (FALSE) |
| ISBASIC | VARCHAR2(5) | Indicates whether the parameter is a basic parameter (TRUE) or not (FALSE) |
| DESCRIPTION | VARCHAR2(255) | Description of the parameter |
| UPDATE_COMMENT | VARCHAR2(255) | Comments associated with the most recent update |
| HASH | NUMBER | Hash value for the parameter name |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$SYSTEM_PARAMETER2

V$SYSTEM_PARAMETER2 displays information about the initialization parameters that are currently in effect for the instance, with each list parameter value appearing as a row in the view. A new session inherits parameter values from the instance-wide values.

Presenting the list parameter values in this format enables you to quickly determine the values for a list parameter. For example, if a parameter value is a, b, then the V$SYSTEM_PARAMETER view does not tell you if the parameter has two values (both a and b) or one value (a, b). V$SYSTEM_PARAMETER2 makes the distinction between the list parameter values clear.

| Column | Datatype | Description |
|---|---|---|
| NUM | NUMBER | Parameter number |
| NAME | VARCHAR2(80) | Name of the parameter |
| TYPE | NUMBER | Parameter type:<br>• 1 - Boolean<br>• 2 - String<br>• 3 - Integer<br>• 4 - Parameter file<br>• 5 - Reserved<br>• 6 - Big integer |
| VALUE | VARCHAR2(4000) | Parameter value |
| DISPLAY_VALUE | VARCHAR2(4000) | Parameter value in a user-friendly format. For example, if the VALUE column shows the value 262144 for a big integer parameter, then the DISPLAY_VALUE column will show the value 256K. |
| ISDEFAULT | VARCHAR2(6) | Indicates whether the parameter is set to the default value (TRUE) or the parameter value was specified in the parameter file (FALSE) |
| ISSES_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be changed with ALTER SESSION (TRUE) or not (FALSE) |
| ISSYS_MODIFIABLE | VARCHAR2(9) | Indicates whether the parameter can be changed with ALTER SYSTEM and when the change takes effect:<br>• IMMEDIATE - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect immediately.<br>• DEFERRED - Parameter can be changed with ALTER SYSTEM regardless of the type of parameter file used to start the instance. The change takes effect in subsequent sessions.<br>• FALSE - Parameter cannot be changed with ALTER SYSTEM unless a server parameter file was used to start the instance. The change takes effect in subsequent instances. |
| ISPDB_MODIFIABLE | VARCHAR2(5) | Indicates whether the parameter can be modified on a per-PDB basis (TRUE) or not (FALSE).<br>In a non-CDB, the value of this column is NULL. |
| ISINSTANCE_MODIFIABLE | VARCHAR2(5) | For parameters that can be changed with ALTER SYSTEM, indicates whether the value of the parameter can be different for every instance (TRUE) or whether the parameter must have the same value for all Real Application Clusters instances (FALSE). If the ISSYS_MODIFIABLE column is FALSE, then this column is always FALSE. |
| ISMODIFIED | VARCHAR2(8) | Indicates how the parameter was modified. If an ALTER SYSTEM was performed, the value will be MODIFIED. |
| ISADJUSTED | VARCHAR2(5) | Indicates whether Oracle adjusted the input value to a more suitable value (for example, the parameter value should be prime, but the user input a non-prime number, so Oracle adjusted the value to the next prime number) |

| Column | Datatype | Description |
|---|---|---|
| ISDEPRECATED | VARCHAR2(5) | Indicates whether the parameter has been deprecated (TRUE) or not (FALSE) |
| ISBASIC | VARCHAR2(5) | Indicates whether the parameter is a basic parameter (TRUE) or not (FALSE) |
| DESCRIPTION | VARCHAR2(255) | Description of the parameter |
| ORDINAL | NUMBER | Position (ordinal number) of the parameter value. Useful only for parameters whose values are lists of strings. |
| UPDATE_COMMENT | VARCHAR2(255) | Comments associated with the most recent update |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$SYSTEM_PARAMETER"

# V$SYSTEM_WAIT_CLASS

V$SYSTEM_WAIT_CLASS displays the instance-wide time totals for each registered wait class.

| Column | Datatype | Description |
|---|---|---|
| WAIT_CLASS_ID | NUMBER | Identifier of the wait class |
| WAIT_CLASS# | NUMBER | Number of the wait class |
| WAIT_CLASS | VARCHAR2(64) | Name of the wait class |
| TOTAL_WAITS | NUMBER | Number of times waits of the class occurred |
| TIME_WAITED | NUMBER | Amount of time (in hundredths of a second) spent in the wait by all sessions in the instance |
| TOTAL_WAITS_FG | NUMBER | Number of times waits from this wait class occurred in foreground sessions |
| TIME_WAITED_FG | NUMBER | Amount of time (in hundredths of a second) spent in waits from this wait class in foreground sessions |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

This view returns instance-wide data and a value of `0` in the `CON_ID` column when queried from the root of a CDB.

# V$TABLESPACE

`V$TABLESPACE` displays tablespace information from the control file.

| Column | Datatype | Description |
|---|---|---|
| `TS#` | `NUMBER` | Tablespace number |
| `NAME` | `VARCHAR2(30)` | Tablespace name |
| `INCLUDED_IN_DATABASE_BACKUP` | `VARCHAR2(3)` | Indicates whether the tablespace is included in full database backups using the `BACKUP DATABASE` RMAN command (`YES`) or not (`NO`); `NO` only if the `CONFIGURE EXCLUDE` RMAN command was used for this tablespace |
| `BIGFILE` | `VARCHAR2(3)` | Indicates whether the tablespace is a bigfile tablespace (`YES`) or a smallfile tablespace (`NO`) |
| `FLASHBACK_ON` | `VARCHAR2(3)` | Indicates whether the tablespace participates in `FLASHBACK DATABASE` operations (`YES`) or not (`NO`) |
| `ENCRYPT_IN_BACKUP` | `VARCHAR2(3)` | Indicates whether encryption is turned ON or off at the tablespace level:<br>• `ON` - Encryption is turned ON at the tablespace level<br>• `OFF` - Encryption is turned OFF at the tablespace level<br>• NULL - Encryption is neither explicitly turned on nor off at the tablespace level (default or when cleared) |
| `CON_ID` | `NUMBER` | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TEMP_CACHE_TRANSFER

`V$TEMP_CACHE_TRANSFER` is deprecated. The information that was provided in this view is now provided in the `V$INSTANCE_CACHE_TRANSFER` and `V$SEGMENT_STATISTICS` views.

| Column | Datatype | Description |
|---|---|---|
| `FILE_NUMBER` | `NUMBER` | Number of the temp file |

| Column | Datatype | Description |
|---|---|---|
| X_2_NULL | NUMBER | Number of blocks with Exclusive-to-NULL conversions; always 0 |
| X_2_NULL_FORCED_WRITE | NUMBER | Number of Exclusive-to-NULL forced writes; always 0 |
| X_2_NULL_FORCED_STALE | NUMBER | Number of Exclusive-to-NULL blocks converted to CR; always 0 |
| X_2_S | NUMBER | Number of blocks with Exclusive-to-Shared conversions; always 0 |
| X_2_S_FORCED_WRITE | NUMBER | Number of Exclusive-to-Shared forced writes; always 0 |
| S_2_NULL | NUMBER | Number of blocks with Shared-to-NULL conversions; always 0 |
| S_2_NULL_FORCED_STALE | NUMBER | Number of Shared-to-NULL blocks converted to CR; always 0 |
| RBR | NUMBER | Number of reuse blocks cross-instance calls; always 0 |
| RBR_FORCED_WRITE | NUMBER | Number of blocks written due to reuse blocks cross-instance calls; always 0 |
| NULL_2_X | NUMBER | Number of blocks with NULL-to-Exclusive conversions; always 0 |
| S_2_X | NUMBER | Number of blocks with Shared-to-Exclusive conversions; always 0 |
| NULL_2_S | NUMBER | Number of blocks with NULL-to-Shared conversions; always 0 |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$INSTANCE_CACHE_TRANSFER"
- "V$SEGMENT_STATISTICS"

# V$TEMP_EXTENT_MAP

V$TEMP_EXTENT_MAP displays the status of each unit for all LOCALLY MANAGED temporary tablespaces.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace this unit belongs to |

| Column | Datatype | NULL | Description |
|---|---|---|---|
| FILE_ID | NUMBER | | Absolute file number |
| BLOCK_ID | NUMBER | | Begin block number for this unit |
| BYTES | NUMBER | | Bytes in the extent |
| BLOCKS | NUMBER | | Blocks in the extent |
| OWNER | NUMBER | | Instance which owns this unit |
| RELATIVE_FNO | NUMBER | | Relative file number |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TEMP_EXTENT_POOL

V$TEMP_EXTENT_POOL displays the state of temporary space cached and used for the instance. Note that loading of the temporary space cache is lazy and that instances can be dormant.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the tablespace |
| FILE_ID | NUMBER | | Absolute file number |
| EXTENTS_CACHED | NUMBER | | Number of extents that have been cached |
| EXTENTS_USED | NUMBER | | Number of extents that are actually being used |
| BLOCKS_CACHED | NUMBER | | Number of blocks that are cached |
| BLOCKS_USED | NUMBER | | Number of blocks that are used |
| BYTES_CACHED | NUMBER | | Number of bytes that are cached |
| BYTES_USED | NUMBER | | Number of bytes that are used |
| RELATIVE_FNO | NUMBER | | Relative file number |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TEMP_SPACE_HEADER

V$TEMP_SPACE_HEADER displays aggregate information per file per LOCALLY MANAGED temporary tablespace regarding how much space is currently being used and how much is free as identified in the space header.

| Column | Datatype | NULL | Description |
|---|---|---|---|
| TABLESPACE_NAME | VARCHAR2(30) | NOT NULL | Name of the temporary tablespace |
| FILE_ID | NUMBER | | Absolute file number |
| BYTES_USED | NUMBER | | How many bytes are in use |
| BLOCKS_USED | NUMBER | | How many blocks are in use |
| BYTES_FREE | NUMBER | | How many bytes are free |
| BLOCKS_FREE | NUMBER | | How many blocks are free |
| RELATIVE_FNO | NUMBER | | The relative file number for the file |
| CON_ID | NUMBER | | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TEMPFILE

V$TEMPFILE displays temp file information.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Absolute file number |
| CREATION_CHANGE# | NUMBER | Creation System Change Number (SCN) |
| CREATION_TIME | DATE | Creation time |
| TS# | NUMBER | Tablespace number |
| RFILE# | NUMBER | Relative file number in the tablespace |
| STATUS | VARCHAR2(7) | Status of the file (OFFLINE\|ONLINE) |
| ENABLED | VARCHAR2(10) | Enabled for read and/or write |
| BYTES | NUMBER | Size of the file in bytes (from the file header) |
| BLOCKS | NUMBER | Size of the file in blocks (from the file header) |
| CREATE_BYTES | NUMBER | Creation size of the file (in bytes) |

| Column | Datatype | Description |
|---|---|---|
| BLOCK_SIZE | NUMBER | Block size for the file |
| NAME | VARCHAR2(513) | Name of the file |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TEMPORARY_LOBS

V$TEMPORARY_LOBS displays information about temporary and abstract LOBs.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID |
| CACHE_LOBS | NUMBER | Number of cache temp LOBs |
| NOCACHE_LOBS | NUMBER | Number of nocache temp LOBs |
| ABSTRACT_LOBS | NUMBER | Number of abstract LOBs |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TEMPSEG_USAGE

V$TEMPSEG_USAGE describes temporary segment usage.

| Column | Datatype | Description |
|---|---|---|
| USERNAME | VARCHAR2(30) | User who requested temporary space |
| USER | VARCHAR2(30) | This column is obsolete and maintained for backward compatibility. The value of this column is always equal to the value in USERNAME. |
| SESSION_ADDR | RAW(4 \| 8) | Session address |
| SESSION_NUM | NUMBER | Serial number of session |
| SQLADDR | RAW(4 \| 8) | Address of SQL statement |
| SQLHASH | NUMBER | Hash value of SQL statement |
| SQL_ID | VARCHAR2(13) | SQL identifier of SQL statement |

| Column | Datatype | Description |
|---|---|---|
| TABLESPACE | VARCHAR2(30) | Tablespace in which space is allocated |
| CONTENTS | VARCHAR2(9) | Indicates whether tablespace is TEMPORARY or PERMANENT |
| SEGTYPE | VARCHAR2(9) | Type of sort segment:<br>• SORT<br>• HASH<br>• DATA<br>• INDEX<br>• LOB_DATA<br>• LOB_INDEX |
| SEGFILE# | NUMBER | File number of initial extent |
| SEGBLK# | NUMBER | Block number of the initial extent |
| EXTENTS | NUMBER | Extents allocated to the sort |
| BLOCKS | NUMBER | Extents in blocks allocated to the sort |
| SEGRFNO# | NUMBER | Relative file number of initial extent |
| TS# | NUMBER | Tablespace number |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| SQL_ID_TEMPSEG | VARCHAR2(13) | SQL identifier of the SQL statement that created the temporary segment |

# V$TEMPSTAT

V$TEMPSTAT displays information about file read/write statistics.

| Column | Datatype | Description |
|---|---|---|
| FILE# | NUMBER | Number of the file |
| PHYRDS | NUMBER | Number of physical reads done |
| PHYWRTS | NUMBER | Number of times DBWR is required to write |
| PHYBLKRD | NUMBER | Number of physical blocks read |
| PHYBLKWRT | NUMBER | Number of blocks written to disk, which may be the same as PHYWRTS if all writes are single blocks |
| SINGLEBLKRDS | NUMBER | Number of single block reads |
| READTIM | NUMBER | Time (in hundredths of a second) spent doing reads if the TIMED_STATISTICS parameter is true; 0 if false |

| Column | Datatype | Description |
|---|---|---|
| WRITETIM | NUMBER | Time (in hundredths of a second) spent doing writes if the TIMED_STATISTICS parameter is true; 0 if false |
| SINGLEBLKRDTIM | NUMBER | Cumulative single block read time (in hundredths of a second) |
| AVGIOTIM | NUMBER | Average time (in hundredths of a second) spent on I/O, if the TIMED_STATISTICS parameter is true; 0 if false |
| LSTIOTIM | NUMBER | Time (in hundredths of a second) spent doing the last I/O, if the TIMED_STATISTICS parameter is true; 0 if false |
| MINIOTIM | NUMBER | Minimum time (in hundredths of a second) spent on a single I/O, if the TIMED_STATISTICS parameter is true; 0 if false |
| MAXIORTM | NUMBER | Maximum time (in hundredths of a second) spent doing a single read, if the TIMED_STATISTICS parameter is true; 0 if false |
| MAXIOWTM | NUMBER | Maximum time (in hundredths of a second) spent doing a single write, if the TIMED_STATISTICS parameter is true; 0 if false |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"TIMED_STATISTICS"

# V$TEMPUNDOSTAT

V$TEMPUNDOSTAT shows various statistics related to the temporary undo log for this database instance. It displays a histogram of statistical data to show how the system is working. Each row in the view keeps statistics collected in the instance for a 10-minute interval. The rows are in the descending order of the BEGIN_TIME column value. This view contains a total of 576 rows, spanning a 4-day cycle. This view is similar to the V$UNDOSTAT view.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Identifies the beginning of the time interval |
| END_TIME | DATE | Identifies the end of the time interval |
| UNDOTSN | NUMBER | Represents the last active undo tablespace in the duration of time. The tablespace ID of the active undo tablespace is returned in this column. If more than one undo tablespace was active in that period, the active undo tablespace that was active at the end of the period is reported. |
| TXNCOUNT | NUMBER | Total number of transaction that have bound to the temp undo segment contained in above tablespace within the interval period |

| Column | Datatype | Description |
|---|---|---|
| MAXCONCURRENCY | NUMBER | Highest number of transactions executed concurrently which modified temporary objects within the interval period |
| MAXQUERYLEN | NUMBER | Reserved for future use |
| MAXQUERYID | VARCHAR2(13) | Reserved for future use |
| UNDOBLKCNT | NUMBER | Total number of temporary undo blocks consumed |
| EXTCNT | NUMBER | Total number of extents consumed |
| USCOUNT | NUMBER | Temp undo segments created in this period |
| SSOLDERRCNT | NUMBER | Identifies the number of times the error ORA-01555 occurred. You can use this statistic to decide whether or not the UNDO_RETENTION initialization parameter is set properly given the size of the undo tablespace. Increasing the value of UNDO_RETENTION can reduce the occurrence of this error. |
| NOSPACEERRCNT | NUMBER | Total number of times the error 'no space left for temporary undo' was raised |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$UNDOSTAT"

# V$THREAD

V$THREAD displays thread information from the control file. This view does not return meaningful results on a physical standby database.

| Column | Datatype | Description |
|---|---|---|
| THREAD# | NUMBER | Thread number |
| STATUS | VARCHAR2(6) | Thread status (OPEN\| CLOSED) |
| ENABLED | VARCHAR2(8) | Enabled status: DISABLED, (enabled) PRIVATE, or (enabled) PUBLIC |
| GROUPS | NUMBER | Number of log groups assigned to this thread |
| INSTANCE | VARCHAR2(80) | Instance name, if available |
| OPEN_TIME | DATE | Last time the thread was opened |
| CURRENT_GROUP# | NUMBER | Current log group |

| Column | Datatype | Description |
|---|---|---|
| SEQUENCE# | NUMBER | Sequence number of current log |
| CHECKPOINT_CHANGE# | NUMBER | SCN at last checkpoint |
| CHECKPOINT_TIME | DATE | Time of last checkpoint |
| ENABLE_CHANGE# | NUMBER | SCN at which thread was enabled |
| ENABLE_TIME | DATE | Time of enable SCN |
| DISABLE_CHANGE# | NUMBER | SCN at which thread was disabled |
| DISABLE_TIME | DATE | Time of disable SCN |
| LAST_REDO_SEQUENCE# | NUMBER | Last redo sequence number written by LGWR |
| LAST_REDO_BLOCK | NUMBER | Last redo block written by LGWR |
| LAST_REDO_CHANGE# | NUMBER | SCN of last redo for the thread |
| LAST_REDO_TIME | DATE | Time of last redo for the thread |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$THRESHOLD_TYPES

V$THRESHOLD_TYPES displays information about threshold types.

| Column | Datatype | Description |
|---|---|---|
| METRICS_ID | NUMBER | Metrics ID |
| METRICS_GROUP_ID | NUMBER | Metrics group ID |
| OPERATOR_MASK | NUMBER | Operator mask |
| OBJECT_TYPE | VARCHAR2(64) | Object type:<br>• SYSTEM<br>• FILE<br>• SERVICE<br>• EVENT_CLASS<br>• TABLESPACE<br>• SESSION |
| ALERT_REASON_ID | NUMBER | ID of the alert reason |
| METRIC_VALUE_TYPE | NUMBER | Metric value type |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TIMER

V$TIMER displays the elapsed time in hundredths of a second. Time is measured since the beginning of the epoch, which is operating system specific, and wraps around to 0 again whenever the value overflows four bytes (roughly 497 days).

| Column | Datatype | Description |
|---|---|---|
| HSECS | NUMBER | Elapsed time (in hundredths of a second) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TIMEZONE_NAMES

V$TIMEZONE_NAMES displays valid time zone names.

| Column | Datatype | Description |
|---|---|---|
| TZNAME | VARCHAR2(64) | Time zone region (for example, US/Pacific) |
| TZABBREV | VARCHAR2(64) | Corresponding daylight abbreviation (for example, PDT) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TOPLEVELCALL

V$TOPLEVELCALL displays the mapping between Oracle top level calls and names.

| Column | Datatype | Description |
|---|---|---|
| TOP_LEVEL_CALL# | NUMBER | Oracle top level call number |
| TOP_LEVEL_CALL_NAME | VARCHAR2(64) | Oracle top level call name |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TRANSACTION

V$TRANSACTION lists the active transactions in the system.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of the transaction state object |
| XIDUSN | NUMBER | Undo segment number |
| XIDSLOT | NUMBER | Slot number |
| XIDSQN | NUMBER | Sequence number |
| UBAFIL | NUMBER | Undo block address (UBA) filenum |
| UBABLK | NUMBER | UBA block number |
| UBASQN | NUMBER | UBA sequence number |
| UBAREC | NUMBER | UBA record number |
| STATUS | VARCHAR2(16) | Status |
| START_TIME | VARCHAR2(20) | Start time (wall clock) |
| START_SCNB | NUMBER | Start system change number (SCN) base |
| START_SCNW | NUMBER | Start SCN wrap |
| START_UEXT | NUMBER | Start extent number |
| START_UBAFIL | NUMBER | Start UBA file number |
| START_UBABLK | NUMBER | Start UBA block number |
| START_UBASQN | NUMBER | Start UBA sequence number |
| START_UBAREC | NUMBER | Start UBA record number |
| SES_ADDR | RAW(4 \| 8) | User session object address |
| FLAG | NUMBER | Flag |
| SPACE | VARCHAR2(3) | YES if a space transaction |
| RECURSIVE | VARCHAR2(3) | YES if a recursive transaction |
| NOUNDO | VARCHAR2(3) | YES if a no undo transaction |

| Column | Datatype | Description |
|---|---|---|
| PTX | VARCHAR2(3) | YES if parallel transaction |
| NAME | VARCHAR2(256) | Name of a named transaction |
| PRV_XIDUSN | NUMBER | Previous transaction undo segment number |
| PRV_XIDSLT | NUMBER | Previous transaction slot number |
| PRV_XIDSQN | NUMBER | Previous transaction sequence number |
| PTX_XIDUSN | NUMBER | Rollback segment number of the parent XID |
| PTX_XIDSLT | NUMBER | Slot number of the parent XID |
| PTX_XIDSQN | NUMBER | Sequence number of the parent XID |
| DSCN-B | NUMBER | This column is obsolete and maintained for backward compatibility. The value of this column is always equal to the value in DSCN_BASE. |
| DSCN-W | NUMBER | This column is obsolete and maintained for backward compatibility. The value of this column is always equal to the value in DSCN_WRAP. |
| USED_UBLK | NUMBER | Number of undo blocks used |
| USED_UREC | NUMBER | Number of undo records used |
| LOG_IO | NUMBER | Logical I/O |
| PHY_IO | NUMBER | Physical I/O |
| CR_GET | NUMBER | Consistent gets |
| CR_CHANGE | NUMBER | Consistent changes |
| START_DATE | DATE | Start time (wall clock) |
| DSCN_BASE | NUMBER | Dependent SCN base |
| DSCN_WRAP | NUMBER | Dependent SCN wrap |
| START_SCN | NUMBER | Start SCN |
| DEPENDENT_SCN | NUMBER | Dependent SCN |
| XID | RAW(8) | Transaction XID |
| PRV_XID | RAW(8) | Previous transaction XID |
| PTX_XID | RAW(8) | Parent transaction XID |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TRANSACTION_ENQUEUE

V$TRANSACTION_ENQUEUE displays locks owned by transaction state objects.

| Column | Datatype | Description |
|---|---|---|
| ADDR | RAW(4 \| 8) | Address of lock state object |
| KADDR | RAW(4 \| 8) | Address of lock |
| SID | NUMBER | Identifier for session holding or acquiring the lock |
| TYPE | VARCHAR2(2) | Type of lock. TX indicates transaction enqueue. |
| ID1 | NUMBER | Lock identifier #1 (depends on type) |
| ID2 | NUMBER | Lock identifier #2 (depends on type) |
| LMODE | NUMBER | Lock mode in which the session holds the lock:<br>• 0 - none<br>• 1 - null (NULL)<br>• 2 - row-S (SS)<br>• 3 - row-X (SX)<br>• 4 - share (S)<br>• 5 - S/Row-X (SSX)<br>• 6 - exclusive (X) |
| REQUEST | NUMBER | Lock mode in which the process requests the lock:<br>• 0 - none<br>• 1 - null (NULL)<br>• 2 - row-S (SS)<br>• 3 - row-X (SX)<br>• 4 - share (S)<br>• 5 - S/Row-X (SSX)<br>• 6 - exclusive (X) |
| CTIME | NUMBER | Time since current mode was granted |
| BLOCK | NUMBER | The lock is blocking another lock |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TRANSPORTABLE_PLATFORM

V$TRANSPORTABLE_PLATFORM displays all platforms that support cross-platform tablespace transport. Specifically, it lists all platforms supported by the RMAN CONVERT TABLESPACE command, along with the endianness of each platform.

| Column | Datatype | Description |
|---|---|---|
| PLATFORM_ID | NUMBER | Platform identification number |
| PLATFORM_NAME | VARCHAR2(101) | Platform name |
| ENDIAN_FORMAT | VARCHAR2(14) | Platform endian format:<br>• Big<br>• Little<br>• UNKNOWN FORMAT |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$TSDP_SUPPORTED_FEATURE

V$TSDP_SUPPORTED_FEATURE displays information about the features supported by Transparent Sensitive Data Protection (TSDP).

| Column | Datatype | Description |
|---|---|---|
| FEATURE_NAME | VARCHAR2(200) | The name of the supported feature |
| FUNCTIONALITY | VARCHAR2(200) | The functionality that is supported within the feature. If all of the functionality of the feature is supported, the value is ALL, otherwise the value will show the specific functionality that is supported. |
| COMMENTS$ | VARCHAR2(4000) | More information regarding the support for the feature and the specific functionality |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

*Oracle Database Security Guide* for more information about using Transparent Sensitive Data Protection.

# V$TYPE_SIZE

V$TYPE_SIZE displays the sizes of various database components for use in estimating data block capacity.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT | VARCHAR2(8) | Component name, such as segment or buffer header |
| TYPE | VARCHAR2(8) | Component type |
| DESCRIPTION | VARCHAR2(32) | Description of the component |
| TYPE_SIZE | NUMBER | Size of the component |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$UNDOSTAT

V$UNDOSTAT displays a histogram of statistical data to show how well the system is working. The available statistics include undo space consumption, transaction concurrency, and length of queries executed in the instance. You can use this view to estimate the amount of undo space required for the current workload. Oracle uses this view to tune undo usage in the system. The view returns NULL values if the system is in manual undo management mode.

Each row in the view keeps statistics collected in the instance for a 10-minute interval. The rows are in descending order by the BEGIN_TIME column value. Each row belongs to the time interval marked by (BEGIN_TIME, END_TIME). Each column represents the data collected for the particular statistic in that time interval. The first row of the view contains statistics for the (partial) current time period. The view contains a total of 576 rows, spanning a 4 day cycle.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Identifies the beginning of the time interval |
| END_TIME | DATE | Identifies the end of the time interval |
| UNDOTSN | NUMBER | Represents the last active undo tablespace in the duration of time. The tablespace ID of the active undo tablespace is returned in this column. If more than one undo tablespace was active in that period, the active undo tablespace that was active at the end of the period is reported. |
| UNDOBLKS | NUMBER | Represents the total number of undo blocks consumed. You can use this column to obtain the consumption rate of undo blocks, and thereby estimate the size of the undo tablespace needed to handle the workload on your system. |
| TXNCOUNT | NUMBER | Identifies the total number of transactions executed within the period |
| MAXQUERYLEN | NUMBER | Identifies the length of the longest query (in seconds) executed in the instance during the period. You can use this statistic to estimate the proper setting of the UNDO_RETENTION initialization parameter. The length of a query is measured from |

| Column | Datatype | Description |
|---|---|---|
| | | the cursor open time to the last fetch/execute time of the cursor. Only the length of those cursors that have been fetched/executed during the period are reflected in the view. |
| MAXQUERYID | VARCHAR2(13) | SQL identifier of the longest running SQL statement in the period |
| MAXCONCURRENCY | NUMBER | Identifies the highest number of transactions executed concurrently within the period |
| UNXPSTEALCNT | NUMBER | Number of attempts to obtain undo space by stealing unexpired extents from other transactions |
| UNXPBLKRELCNT | NUMBER | Number of unexpired blocks removed from certain undo segments so they can be used by other transactions |
| UNXPBLKREUCNT | NUMBER | Number of unexpired undo blocks reused by transactions |
| EXPSTEALCNT | NUMBER | Number of attempts to steal expired undo blocks from other undo segments |
| EXPBLKRELCNT | NUMBER | Number of expired undo blocks stolen from other undo segments |
| EXPBLKREUCNT | NUMBER | Number of expired undo blocks reused within the same undo segments |
| SSOLDERRCNT | NUMBER | Identifies the number of times the error `ORA-01555` occurred. You can use this statistic to decide whether or not the `UNDO_RETENTION` initialization parameter is set properly given the size of the undo tablespace. Increasing the value of `UNDO_RETENTION` can reduce the occurrence of this error. |
| NOSPACEERRCNT | NUMBER | Identifies the number of times space was requested in the undo tablespace and there was no free space available. That is, all of the space in the undo tablespace was in use by active transactions. The corrective action is to add more space to the undo tablespace. |
| ACTIVEBLKS | NUMBER | Total number of blocks in the active extents of the undo tablespace for the instance at the sampled time in the period |
| UNEXPIREDBLKS | NUMBER | Total number of blocks in the unexpired extents of the undo tablespace for the instance at the sampled time in the period |
| EXPIREDBLKS | NUMBER | Total number of blocks in the expired extents of the undo tablespace for the instance at the sampled time in the period |
| TUNED_UNDORETENTION | NUMBER | Amount of time (in seconds) for which undo will not be recycled from the time it was committed. At any point in time, the latest value of `TUNED_UNDORETENTION` is used to determine whether data committed at a particular time in the past can be recycled. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• `0`: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• `1`: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"UNDO_RETENTION"

# V$UNUSABLE_BACKUPFILE_DETAILS

V$UNUSABLE_BACKUPFILE_DETAILS displays information about all backup files (backup pieces, proxy copies, or copies) that are marked unavailable and expired. You can select one of the rows and use BTYPE_KEY or FILETYPE_KEY to change the status of a backup file set or a specific file to available.

| Column | Datatype | Description |
|---|---|---|
| SESSION_KEY | NUMBER | Session identifier |
| SESSION_RECID | NUMBER | Session record ID |
| SESSION_STAMP | NUMBER | Session stamp |
| RMAN_STATUS_RECID | NUMBER | Record ID of the corresponding row in the control file |
| RMAN_STATUS_STAMP | NUMBER | Timestamp of the row in the controlfile |
| BTYPE | CHAR(9) | Backup type container. Possible values are: BACKUPSET, IMAGECOPY, PROXYCOPY. |
| BTYPE_KEY | NUMBER | Unique identifier for the backup type, either BS_KEY or COPY_KEY. |
| ID1 | NUMBER | If BACKUPSET, it contains SET_STAMP.<br>If IMAGECOPY or PROXYCOPY, it is RECID from the control file. |
| ID2 | NUMBER | If BACKUPSET, it contains SET_COUNT.<br>If IMAGECOPY or PROXYCOPY, it is STAMP. |
| FILETYPE | VARCHAR2(15) | Type of file. Possible values are: BACKUPPIECE, COPY, PROXYCOPY. |
| FILETYPE_KEY | NUMBER | Backup piece key if the file is a backup piece; otherwise COPY_KEY. |
| STATUS | VARCHAR2(1) | Status of the backup file, either U (unavailable) or X (expired) |
| FILESIZE | NUMBER | Size of the file |
| DEVICE_TYPE | VARCHAR2(17) | Type of device on which the file resides |
| FILENAME | VARCHAR2(513) | Name of the file |
| MEDIA | VARCHAR2(65) | Name of the media on which the copy resides. This value is informational only. It is not needed for restore. |
| MEDIA_POOL | NUMBER | Media pool in which the copy resides. This is the same value that was entered in the POOL operand of the Recovery Manager BACKUP command. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include: |

| Column | Datatype | Description |
|---|---|---|
| | | • 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$VERSION

V$VERSION displays version numbers of core library components in the Oracle Database. There is one row for each component.

| Column | Datatype | Description |
|---|---|---|
| BANNER | VARCHAR2(80) | Component name and version number |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$VPD_POLICY

V$VPD_POLICY displays all the fine-grained security policies and predicates associated with the cursors currently in the library cache.

| Column | Datatype | Description |
|---|---|---|
| ADDRESS | RAW(4 \| 8) | Cursor address |
| PARADDR | RAW(4 \| 8) | Parent cursor address |
| SQL_HASH | NUMBER | SQL hash number |
| SQL_ID | VARCHAR2(13) | SQL identifier |
| CHILD_NUMBER | NUMBER | Cursor's child number under the parent |
| OBJECT_OWNER | VARCHAR2(30) | Owner of the object with the policy |
| OBJECT_NAME | VARCHAR2(30) | Name of the object with the policy |
| POLICY_GROUP | VARCHAR2(30) | Name of the policy group |
| POLICY | VARCHAR2(30) | Name of the policy |
| POLICY_FUNCTION_OWNER | VARCHAR2(30) | Owner of the policy function |
| PREDICATE | VARCHAR2(4000) | Predicate for the policy (truncated to 4000 bytes in length) |

| Column | Datatype | Description |
|---|---|---|
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$WAIT_CHAINS

V$WAIT_CHAINS displays information about blocked sessions. A wait chain is composed of sessions that are blocked by one another. Each row represents a blocked and blocker session pair. If a wait chain is not a cyclical wait chain, then the last row for the chain does not have a blocker.

| Column | Datatype | Description |
|---|---|---|
| CHAIN_ID | NUMBER | A number identifying the wait chain |
| CHAIN_IS_CYCLE | VARCHAR2(5) | Indicates whether the final blocked session in the wait chain is blocked by the initial blocked session (TRUE) or not (FALSE) |
| CHAIN_SIGNATURE | VARCHAR2(801) | An Oracle-specific text signature of the wait chain. This signature can be used to identify similar wait chains. |
| CHAIN_SIGNATURE_HASH | NUMBER | A numeric representation of CHAIN_SIGNATURE |
| INSTANCE | NUMBER | Blocked session's instance identifier |
| OSID | VARCHAR2(25) | Blocked session's operating system process identifier |
| PID | NUMBER | Blocked session's Oracle process identifier |
| SID | NUMBER | Blocked session's Oracle session identifier |
| SESS_SERIAL# | NUMBER | Blocked session's Oracle session serial number |
| PDB_ID | NUMBER | Blocked session's PDB identifier |
| PDB_NAME | VARCHAR2(31) | Blocked session's PDB name |
| BLOCKER_IS_VALID | VARCHAR2(5) | Indicates whether the blocked session has a blocker (TRUE) or not (FALSE) |
| BLOCKER_INSTANCE | NUMBER | Blocker session's instance identifier; NULL if BLOCKER_IS_VALID is FALSE |
| BLOCKER_OSID | VARCHAR2(25) | Blocker session's operating system process identifier; NULL if BLOCKER_IS_VALID is FALSE |
| BLOCKER_PID | NUMBER | Blocker session's Oracle process identifier; NULL if BLOCKER_IS_VALID is FALSE |
| BLOCKER_SID | NUMBER | Blocker session's Oracle session identifier; NULL if BLOCKER_IS_VALID is FALSE |

| Column | Datatype | Description |
|---|---|---|
| BLOCKER_SESS_SERIAL# | NUMBER | Blocker session's Oracle session serial number; NULL if BLOCKER_IS_VALID is FALSE |
| BLOCKER_PDB_ID | NUMBER | Blocker session's PDB identifier |
| BLOCKER_PDB_NAME | VARCHAR2(31) | Blocker session's PDB name |
| BLOCKER_CHAIN_ID | NUMBER | If not NULL, then the blocker session is a member of another chain specified by this chain identifier. For the remaining wait chain information, see the wait chain with the specified CHAIN_ID. |
| IN_WAIT | VARCHAR2(5) | Indicates whether the blocked session is in a wait (TRUE) or not (FALSE) |
| TIME_SINCE_LAST_WAIT_SECS | NUMBER | Number of seconds since the last time the blocked session waited; NULL if IN_WAIT is TRUE |
| WAIT_ID | NUMBER | A number identifying the wait; NULL if IN_WAIT is FALSE |
| WAIT_EVENT | NUMBER | Resource or event number for which the blocked session is waiting; NULL if IN_WAIT is FALSE |
| WAIT_EVENT_TEXT | VARCHAR2(64) | Resource or event for which the blocked session is waiting; NULL if IN_WAIT is FALSE |
| P1 | NUMBER | First additional wait parameter; NULL if IN_WAIT is FALSE |
| P1_TEXT | VARCHAR2(64) | Description of the first additional wait parameter; NULL if IN_WAIT is FALSE |
| P2 | NUMBER | Second additional wait parameter; NULL if IN_WAIT is FALSE |
| P2_TEXT | VARCHAR2(64) | Description of the second additional wait parameter; NULL if IN_WAIT is FALSE |
| P3 | NUMBER | Third additional wait parameter; NULL if IN_WAIT is FALSE |
| P3_TEXT | VARCHAR2(64) | Description of the third additional wait parameter; NULL if IN_WAIT is FALSE |
| IN_WAIT_SECS | NUMBER | Seconds the blocked session has been in the current wait; NULL if IN_WAIT is FALSE |
| TIME_REMAINING_SECS | NUMBER | Seconds remaining until the blocked session ends its wait (-1 if the blocked session can indefinitely wait); NULL if IN_WAIT is FALSE |
| NUM_WAITERS | NUMBER | Number of sessions waiting for the blocked session |
| ROW_WAIT_OBJ# | NUMBER | Object ID for the table containing the row specified in ROW_WAIT_ROW#; NULL if IN_WAIT is FALSE |
| ROW_WAIT_FILE# | NUMBER | Identifier for the data file containing the row specified in ROW_WAIT_ROW#; NULL if IN_WAIT is FALSE. This column is valid only if the blocked session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is not -1. |

| Column | Datatype | Description |
|---|---|---|
| ROW_WAIT_BLOCK# | NUMBER | Identifier for the block containing the row specified in ROW_WAIT_ROW#; NULL if IN_WAIT is FALSE. This column is valid only if the blocked session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is not -1. |
| ROW_WAIT_ROW# | NUMBER | Current row being locked; NULL if IN_WAIT is FALSE. This column is valid only if the blocked session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is not -1. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$WAITCLASSMETRIC

V$WAITCLASSMETRIC displays metric values of wait classes for the most recent 60-second interval. A history of the last one hour will be kept in the system.

| Column | Datatype | Description |
|---|---|---|
| BEGIN_TIME | DATE | Begin time of the interval |
| END_TIME | DATE | End time of the interval |
| INTSIZE_CSEC | NUMBER | Interval size (in hundredths of a second) |
| WAIT_CLASS# | NUMBER | Number of the class of the wait event |
| WAIT_CLASS_ID | NUMBER | Identifier of the class of the wait event |
| AVERAGE_WAITER_COUNT | NUMBER | Average waiter count |
| DBTIME_IN_WAIT | NUMBER | Percent of database time spent in the wait |
| TIME_WAITED | NUMBER | Time waited during the interval (in hundredths of a second) |
| WAIT_COUNT | NUMBER | Number of times waited |
| TIME_WAITED_FG | NUMBER | Amount of time (in hundredths of a second) spent in waits from this wait class in foreground sessions |
| WAIT_COUNT_FG | NUMBER | Number of times foreground processes waited |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root |

| Column | Datatype | Description |
|---|---|---|
| | | • *n*: Where *n* is the applicable container ID for the rows containing data |

# V$WAITCLASSMETRIC_HISTORY

V$WAITCLASSMETRIC_HISTORY displays metric values of wait classes for all intervals in the last one hour.

The columns for V$WAITCLASSMETRIC_HISTORY are the same as those for V$WAITCLASSMETRIC.

**See Also:**

"V$WAITCLASSMETRIC"

# V$WAITSTAT

V$WAITSTAT displays block contention statistics. This table is only updated when timed statistics are enabled.

| Column | Datatype | Description |
|---|---|---|
| CLASS | VARCHAR2(18) | Class of the block |
| COUNT | NUMBER | Number of waits by this OPERATION for this CLASS of block |
| TIME | NUMBER | Sum of all wait times for all the waits by this OPERATION for this CLASS of block (in centiseconds) |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$WALLET

V$WALLET displays metadata of certificates that may be used as a master key for Transparent Data Encryption.

**Note:**

The use of PKI encryption with Transparent Data Encryption is deprecated. To configure Transparent Data Encryption, use the ADMINISTER KEY MANAGEMENT SQL statement. See *Oracle Database Advanced Security Guide* for more information.

| Column | Datatype | Description |
|---|---|---|
| CERT_ID | VARCHAR2(52) | A unique certificate identifier value used to specify a particular PKI certificate for use as the master key |
| DN | VARCHAR2(255) | Distinguished name of a particular PKI certificate |
| SERIAL_NUM | VARCHAR2(40) | Unique serial number assigned to a certificate by the issuer or signer |
| ISSUER | VARCHAR2(255) | Distinguished name of the Certificate Authority or issuer that issued and signed the certificate |
| KEYSIZE | NUMBER | Size of the PKI key associated with the certificate |
| STATUS | VARCHAR2(16) | Current status of the certificate:<br>• UNUSED<br>• IN USE<br>• USED<br>This column allows the user to identify whether a certificate is currently in use or has already been used for Transparent Data Encryption. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$WORKLOAD_REPLAY_THREAD

V$WORKLOAD_REPLAY_THREAD displays information for all the different types of replay sessions.

| Column | Datatype | Description |
|---|---|---|
| CLOCK | NUMBER | Current simulated SCN based on the SCN recorded at capture time |
| NEXT_TICKER | NUMBER | Next stream ID that will increment the CLOCK after a commit action |
| SID | NUMBER | Session ID of the replay session |
| SERIAL# | NUMBER | Session serial number of the replay session |
| SPID | VARCHAR2(24) | The server process ID of the replay session |
| LOGON_USER | VARCHAR2(30) | Logon username of the replay session |
| LOGON_TIME | DATE | Logon time of the replay session |
| EVENT | VARCHAR2(64) | Event name |
| EVENT_ID | NUMBER | Event identifier |

| Column | Datatype | Description |
|---|---|---|
| EVENT# | NUMBER | Event operation code |
| P1TEXT | VARCHAR2(64) | Text for event parameter 1 |
| P1 | NUMBER | Value of event parameter 1 |
| P2TEXT | VARCHAR2(64) | Text for event parameter 2 |
| P2 | NUMBER | Value of event parameter 2 |
| P3TEXT | VARCHAR2(64) | Text for event parameter 3 |
| P3 | NUMBER | Value of event parameter 3 |
| WAIT_FOR_SCN | NUMBER | The captured SCN for which the current user call should wait |
| FILE_ID | NUMBER | The stream ID that is being replayed |
| CALL_COUNTER | NUMBER | The call counter of the user call that is being replayed |
| DEPENDENT_SCN | NUMBER | The dependent SCN, captured to order the commit actions using block-level dependencies |
| STATEMENT_SCN | NUMBER | Statement SCN |
| COMMIT_WAIT_SCN | NUMBER | The (maximum) SCN that the current commit should wait for |
| POST_COMMIT_SCN | NUMBER | The next SCN after the current commit |
| ACTION_TYPE | NUMBER | The type of commits. The possible values are: COMMIT, ROLLBACK, FAKED_COMMIT, and NULL. A value of NULL means it is not a commit operation. |
| SESSION_TYPE | VARCHAR2(13) | The type of replay session:<br>LOGON<br>ADMIN<br>DISPATCHER<br>REPLAY |
| WRC_ID | NUMBER | Unique replay client ID assigned by the server to all participating replay clients when replay starts |
| FILE_NAME | VARCHAR2(31) | File name of the captured stream |
| SKIP_IT | VARCHAR2(1) | Whether or not the current replayed user call is skipped or not |
| DIRTY_BUFFERS | VARCHAR2(1) | Reserved for internal use |
| DBTIME | NUMBER | Accumulated database time for the replay session |
| NETWORK_TIME | NUMBER | Accumulated network time for the replay session |
| THINK_TIME | NUMBER | Accumulated think time for the replay session |
| TIME_GAIN | NUMBER | If nonzero, the accumulated time in macro seconds indicating how fast the replay is |
| TIME_LOSS | NUMBER | If nonzero, the accumulated time in macro seconds indicating how slow the replay is |

| Column | Datatype | Description |
|---|---|---|
| USER_CALLS | NUMBER | Total number of user calls |
| CLIENT_OS_USER | VARCHAR2(15) | Operating system username of the replay client |
| CLIENT_HOST | VARCHAR2(64) | Host name of the replay client |
| CLIENT_PID | VARCHAR2(24) | Process ID of the replay client |
| PROGRAM | VARCHAR2(48) | Program name of the replay client |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$XML_AUDIT_TRAIL

V$XML_AUDIT_TRAIL shows standard, fine-grained, SYS, and mandatory audit records written in XML format files.

**Note:**

This view is populated only in an Oracle Database where unified auditing is not enabled. When unified auditing is enabled in Oracle Database, the audit records are populated in the new audit trail and can be viewed from UNIFIED_AUDIT_TRAIL.

- See *Oracle Database Security Guide* for more information about unified auditing.
- See *Oracle Database Upgrade Guide* for more information about migrating to unified auditing.

| Column | Datatype | Description |
|---|---|---|
| AUDIT_TYPE | NUMBER | Type of audit row:<br>• 1 = Standard XML Audit<br>• 2 = Fine Grained XML Audit<br>• 4 = SYS XML Audit<br>• 8 = Mandatory XML Audit |
| SESSION_ID | NUMBER | Numeric ID for the Oracle session |
| PROXY_SESSIONID | NUMBER | Proxy session serial number, if an enterprise user has logged in through a proxy mechanism |
| STATEMENTID | NUMBER | Numeric ID for the statement run (a statement may cause multiple audit records) |
| ENTRYID | NUMBER | Numeric ID for the audit trail entry in the session |

| Column | Datatype | Description |
|---|---|---|
| EXTENDED_TIMESTAMP | TIMESTAMP(6) WITH TIME ZONE | Timestamp of the audited operation (the timestamp of the user's logon for entries is created by AUDIT SESSION) |
| GLOBAL_UID | VARCHAR2(32) | Global user identifier for the user, if the user has logged in as an enterprise user |
| DB_USER | VARCHAR2(30) | Database username of the user whose actions were audited |
| CLIENTIDENTIFIER | VARCHAR2(64) | Client identifier in the Oracle session |
| EXT_NAME | VARCHAR2(1024) | User's external name |
| OS_USER | VARCHAR2(30) | Operating system logon user name of the user whose actions were audited |
| OS_HOST | VARCHAR2(128) | Client host system name |
| OS_PROCESS | VARCHAR2(16) | Operating system process identifier of the Oracle server process |
| TERMINAL | VARCHAR2(30) | Identifier for the user's terminal |
| INSTANCE_NUMBER | NUMBER | Instance number as specified by the INSTANCE_NUMBER initialization parameter |
| OBJECT_SCHEMA | VARCHAR2(30) | Owner of the audited object |
| OBJECT_NAME | VARCHAR2(30) | Name of the object affected by the action |
| POLICY_NAME | VARCHAR2(30) | Name of the fine-grained auditing policy |
| NEW_OWNER | VARCHAR2(30) | Owner of the object named in the NEW_NAME column |
| NEW_NAME | VARCHAR2(30) | New name of object after renaming, or the name of an underlying object (for example, CREATE INDEX owner.obj_name ON new_owner.new_name) |
| ACTION | NUMBER | Numeric code for the action type |
| STATEMENT_TYPE | NUMBER | Description of the action |
| TRANSACTIONID | RAW(8) | Identifier of the transaction in which the object is accessed or modified |
| RETURNCODE | NUMBER | Oracle error code generated by the action. Zero if the action succeeded. |
| SCN | NUMBER | System change number (SCN) of the query |
| COMMENT_TEXT | VARCHAR2(4000) | Text comments on standard audit entries. Also indicates how the user was authenticated - the method can be one of the following:<br>• DATABASE - authentication was done by password<br>• NETWORK - authentication was done by Oracle Net Services or the Advanced Networking Option<br>• PROXY - the client was authenticated by another user. The name of the proxy user follows the method type. |
| AUTH_PRIVILEGES | VARCHAR2(16) | Privileges granted and revoked in GRANT and REVOKE statements recorded for standard audit trail entry |

| Column | Datatype | Description |
|---|---|---|
| GRANTEE | VARCHAR2(30) | User who granted or revoked the privilege |
| PRIV_USED | NUMBER | Numerical code of privileges, if any, used in the action |
| SES_ACTIONS | VARCHAR2(16) | Session summary for standard audit records. A string of 12 characters, one for each action type, in the following order: Alter, Audit, Comment, Delete, Grant, Index, Insert, Lock, Rename, Select, Update, Flashback.<br>Values: - = None, S=Success, F=Failure, B=Both |
| OS_PRIVILEGE | VARCHAR2(7) | Operating system privilege (SYSDBA or SYSOPER), if any, used in the session. If no privilege is used, it will be NONE. |
| ECONTEXT_ID | VARCHAR2(64) | Application execution context identifier |
| SQL_BIND | VARCHAR2(4000) | List of bind variables used in the statement |
| SQL_TEXT | VARCHAR2(4000) | The statement or command that triggered the audit event |
| OBJ_EDITION_NAME | VARCHAR2(30) | Name of the edition containing the audited object |
| DBID | NUMBER | Database identifier of the audited database |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The SQL_BIND and SQL_TEXT columns are only populated if the AUDIT_TRAIL initialization parameter is set to xml, extended or if the AUDIT_SYS_OPERATIONS initialization parameter is set to TRUE.

**See Also:**

- "UNIFIED_AUDIT_TRAIL"
- "AUDIT_SYS_OPERATIONS"
- "AUDIT_TRAIL"

# V$XSTREAM_APPLY_COORDINATOR

V$XSTREAM_APPLY_COORDINATOR displays information about each XStream apply process coordinator. The coordinator for an apply process gets transactions from the apply process reader and passes them to apply servers. An apply process coordinator is a subcomponent of an apply process, outbound server, or inbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the coordinator's session |
| SERIAL# | NUMBER | Serial number of the coordinator's session |
| STATE | VARCHAR2(21) | State of the coordinator:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• APPLYING - Passing transactions to apply servers<br>• SHUTTING DOWN CLEANLY - Stopping without an error<br>• ABORTING - Stopping because of an apply error |
| APPLY# | NUMBER | Apply process number. An apply process coordinator is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| TOTAL_APPLIED | NUMBER | Total number of transactions applied by the apply process since the apply process was last started |
| TOTAL_WAIT_DEPS | NUMBER | Number of times since the apply process was last started that an apply server waited to apply a logical change record (LCR) in a transaction until another apply server applied a transaction because of a dependency between the transactions |
| TOTAL_WAIT_COMMITS | NUMBER | Number of times since the apply process was last started that an apply server waited to commit a transaction until another apply server committed a transaction to serialize commits |
| TOTAL_ADMIN | NUMBER | Number of administrative jobs issued since the apply process was last started |
| TOTAL_ASSIGNED | NUMBER | Number of transactions assigned to apply servers since the apply process was last started |
| TOTAL_RECEIVED | NUMBER | Total number of transactions received by the coordinator process since the apply process was last started |
| TOTAL_IGNORED | NUMBER | Number of transactions which were received by the coordinator but were ignored because they had been previously applied |
| TOTAL_ROLLBACKS | NUMBER | Number of transactions which were rolled back due to unexpected contention |
| TOTAL_ERRORS | NUMBER | Number of transactions applied by the apply process that resulted in an apply error since the apply process was last started |
| UNASSIGNED_COMPLETE_TXNS | NUMBER | Total number of complete transactions that the coordinator has not assigned to any apply servers |
| LWM_TIME | DATE | Time when the message with the lowest message number was recorded. The creation time of the message with the lowest message number was also recorded at this time. |
| LWM_MESSAGE_NUMBER | NUMBER | Number of the message corresponding to the low watermark. That is, messages with a commit message number less than or equal to this message number have definitely been applied, but some messages with a higher commit message number also may have been applied. |

| Column | Datatype | Description |
|---|---|---|
| LWM_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the message corresponding to the low watermark. For user-enqueued messages, time when the message corresponding to the low watermark was enqueued into the queue at the local database. |
| HWM_TIME | DATE | Time when the message with the highest message number was recorded. The creation time of the message with the highest message number was also recorded at this time. |
| HWM_MESSAGE_NUMBER | NUMBER | Number of the message corresponding to the high watermark. That is, no messages with a commit message number greater than this message number have been applied. |
| HWM_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the message corresponding to the high watermark. For user-enqueued messages, time when the message corresponding to the high watermark was enqueued into the queue at the local database. |
| STARTUP_TIME | DATE | Time when the apply process was last started |
| ELAPSED_SCHEDULE_TIME | NUMBER | Time elapsed (in hundredths of a second) scheduling messages since the apply process was last started |
| ELAPSED_IDLE_TIME | NUMBER | Elapsed idle time |
| LWM_POSITION | RAW(64) | Position of the low-watermark LCR |
| HWM_POSITION | RAW(64) | Position of the high-watermark LCR |
| PROCESSED_MESSAGE_NUMBER | NUMBER | Message number currently processed by the apply coordinator |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| ACTIVE_SERVER_COUNT[1] | NUMBER | Active server count |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**Note:**

The ELAPSED_SCHEDULE_TIME column is only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$XSTREAM_APPLY_READER

V$XSTREAM_APPLY_READER displays information about each XStream apply reader. The apply reader is a process which reads (dequeues) messages from the queue, computes message dependencies, and builds transactions. It passes the transactions on to the coordinator in commit order for assignment to the apply servers. An apply reader is a subcomponent of an apply process, outbound server, or inbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the reader's session |
| SERIAL# | NUMBER | Serial number of the reader's session |
| APPLY# | NUMBER | Apply process number. An apply process is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| STATE | VARCHAR2(36) | Shows the state of the apply reader and the hash server. The possible values include:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• DEQUEUE MESSAGES - Dequeuing messages from the queue<br>• SCHEDULE MESSAGES - Computing dependencies between messages and assembling messages into transactions<br>• SPILLING - Spilling unapplied messages from memory to hard disk<br>• PAUSED - WAITING FOR DDL TO COMPLETE - Waiting for a data definition language (DDL) logical change record (LCR) to be applied<br>The state of the apply reader is displayed first, followed by the state of the hash server. A semicolon separates the apply reader state from the hash server state. |
| TOTAL_MESSAGES_DEQUEUED | NUMBER | Total number of messages dequeued since the apply process was last started |
| TOTAL_MESSAGES_SPILLED | NUMBER | Number of messages spilled by the reader since the apply process was last started |
| DEQUEUE_TIME | DATE | Time when the last message was received |
| DEQUEUED_MESSAGE_NUMBER | NUMBER | Number of the last message received |
| DEQUEUED_MESSAGE_CREATE_TIME | DATE | For captured messages, creation time at the source database of the last message received. For user-enqueued messages, time when the message was enqueued into the queue at the local database. |

| Column | Datatype | Description |
|---|---|---|
| SGA_USED | NUMBER | Amount (in bytes) of SGA memory used by the apply process since it was last started |
| ELAPSED_DEQUEUE_TIME | NUMBER | Time elapsed (in hundredths of a second) dequeuing messages since the apply process was last started |
| ELAPSED_SCHEDULE_TIME | NUMBER | Time elapsed (in hundredths of a second) scheduling messages since the apply process was last started. Scheduling includes computing dependencies between messages and assembling messages into transactions. |
| ELAPSED_SPILL_TIME | NUMBER | Elapsed time (in hundredths of a second) spent spilling messages since the apply process was last started |
| OLDEST_SCN_NUM | NUMBER | Oldest SCN |
| OLDEST_XIDUSN | NUMBER | Transaction ID undo segment number of the oldest transaction that either has been applied or is being applied |
| OLDEST_XIDSLT | NUMBER | Transaction ID slot number of the oldest transaction that either has been applied or is being applied |
| OLDEST_XIDSQN | NUMBER | Transaction ID sequence number of the oldest transaction that either has been applied or is being applied |
| SPILL_LWM_SCN | NUMBER | Spill low-watermark SCN |
| PROXY_SID | NUMBER | When the apply process uses combined capture and apply, the session ID of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is `0`. |
| PROXY_SERIAL | NUMBER | When the apply process uses combined capture and apply, the serial number of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is `0`. |
| PROXY_SPID | VARCHAR2(12) | When the apply process uses combined capture and apply, the process identification number of the propagation receiver that is responsible for direct communication between capture and apply. If the apply process does not use combined capture and apply, then this column is `0`. |
| BYTES_RECEIVED | NUMBER | When the apply process uses combined capture and apply, the number of bytes received by the apply process from the capture process since the apply process last started. If the apply process does not use combined capture and apply, then this column is not populated. |
| DEQUEUED_POSITION | RAW(64) | Dequeued position. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| SPILL_LWM_POSITION | RAW(64) | Spill low-watermark position. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| OLDEST_TRANSACTION_ID | VARCHAR2(128) | Oldest transaction ID |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_LCRS_WITH_DEP | NUMBER | Total number of LCRs with row-level dependencies since the apply process last started |
| TOTAL_LCRS_WITH_WMDEP | NUMBER | Total number of LCRs with watermark dependencies since the apply process last started.<br>A watermark dependency occurs when an apply process must wait until the apply process's low watermark reaches a particular threshold. |
| TOTAL_IN_MEMORY_LCRS | NUMBER | Total number of LCRs currently in memory |
| SGA_ALLOCATED | NUMBER | The total amount of shared memory (in bytes) allocated from the XStreams pool for the apply process since the apply process last started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The ELAPSED_SCHEDULE_TIME column is only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$XSTREAM_APPLY_RECEIVER

V$XSTREAM_APPLY_RECEIVER displays information about the message receiver of the apply process. The values are reset to zero when the database (or instance in an Oracle Real Application Clusters (Oracle RAC) environment) restarts, when apply migrates to another instance, or when the XStream process is stopped.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the apply receiver |
| SERIAL# | NUMBER | Serial number of the apply receiver |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| STARTUP_TIME | DATE | Startup time of the apply process |

| Column | Datatype | Description |
|---|---|---|
| SOURCE_DATABASE_NAME | VARCHAR2(128) | Name of the source database |
| ACKNOWLEDGEMENT | NUMBER | Acknowledgement SCN of the messages received by the receiver |
| LAST_RECEIVED_MSG | NUMBER | Last received message |
| TOTAL_MESSAGES_RECEIVED | NUMBER | Total number of messages received |
| TOTAL_AVAILABLE_MESSAGES | NUMBER | Number of available messages |
| STATE | VARCHAR2(44) | State of the apply receiver:<br>• Initializing<br>• Sending unapplied txns<br>• Waiting for message from client<br>• Waiting for LCR from client<br>• Receiving LCRs<br>• Evaluating rules<br>• Enqueueing LCRS<br>• Waiting for memory<br>• Waiting for apply to read<br>• Waiting for client flush request to complete<br>• Waiting for client commit to complete |
| LAST_RECEIVED_MSG_POSITION | RAW(64) | Last received message position |
| ACKNOWLEDGEMENT_POSITION | RAW(64) | Acknowledgement position of the messages received by the receiver. Corresponds to ACKNOWLEDGEMENT, except the value is in position rather than SCN. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

# V$XSTREAM_APPLY_SERVER

V$XSTREAM_APPLY_SERVER displays information about each XStream apply server and its activities. An apply server receives messages from the apply coordinator for an apply process. For each message received, an apply server either applies the message or sends the message to the appropriate apply handler. An apply server is a subcomponent of an apply process, outbound server, or inbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the apply server's session |
| SERIAL# | NUMBER | Serial number of the apply server's session |

| Column | Datatype | Description |
|---|---|---|
| APPLY# | NUMBER | Apply process number. An apply process is an Oracle background process, prefixed by ap. |
| APPLY_NAME | VARCHAR2(30) | Name of the apply process |
| SERVER_ID | NUMBER | Parallel execution server number of the apply server |
| STATE | VARCHAR2(20) | State of the apply server:<br>• INITIALIZING - Starting up<br>• IDLE - Performing no work<br>• RECORD LOW-WATERMARK - Performing an administrative job that maintains information about the apply progress, which is used in the ALL_APPLY_PROGRESS and DBA_APPLY_PROGRESS data dictionary views<br>• ADD PARTITION - Performing an administrative job that adds a partition that is used for recording information about in-progress transactions<br>• DROP PARTITION - Performing an administrative job that purges rows that were used to record information about in-progress transactions<br>• EXECUTE TRANSACTION - Applying a transaction<br>• WAIT COMMIT - Waiting to commit a transaction until all other transactions with a lower commit SCN are applied. This state is possible only if the COMMIT_SERIALIZATION apply process parameter is set to a value other than DEPENDENT_TRANSACTIONS and the PARALLELISM apply process parameter is set to a value greater than 1.<br>• WAIT DEPENDENCY - Waiting to apply a logical change record (LCR) in a transaction until another transaction, on which it has a dependency, is applied. This state is possible only if the PARALLELISM apply process parameter is set to a value greater than 1.<br>• ROLLBACK TRANSACTION - Rolling back a transaction<br>• TRANSACTION CLEANUP - Cleaning up an applied transaction, which includes removing LCRs from the apply process's queue<br>• WAIT FOR CLIENT - Waiting for an XStream client application to request more LCRs<br>• WAIT FOR NEXT CHUNK - Waiting for the next set of LCRs for a large transaction |
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction currently being applied |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction currently being applied |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction currently being applied |
| COMMITSCN | NUMBER | Commit system change number (SCN) of the transaction currently being applied |
| DEP_XIDUSN | NUMBER | Transaction ID undo segment number of a transaction on which the transaction being applied by this apply server depends |

| Column | Datatype | Description |
|---|---|---|
| DEP_XIDSLT | NUMBER | Transaction ID slot number of a transaction on which the transaction being applied by this apply server depends |
| DEP_XIDSQN | NUMBER | Transaction ID sequence number of a transaction on which the transaction being applied by this apply server depends |
| DEP_COMMITSCN | NUMBER | Commit system change number (SCN) of the transaction on which this apply server depends |
| MESSAGE_SEQUENCE | NUMBER | Number of the current message being applied by the apply server. This value is reset to 1 at the beginning of each transaction. |
| TOTAL_ASSIGNED | NUMBER | Total number of transactions assigned to the apply server since the apply process was last started |
| TOTAL_ADMIN | NUMBER | Total number of administrative jobs done by the apply server since the apply process was last started. See the STATE information in this view for the types of administrative jobs. |
| TOTAL_ROLLBACKS | NUMBER | Number of transactions assigned to this server which were rolled back |
| TOTAL_MESSAGES_APPLIED | NUMBER | Total number of messages applied by this apply server since the apply process was last started |
| APPLY_TIME | DATE | Time the last message was applied |
| APPLIED_MESSAGE_NUMBER | NUMBER | Number of the last message applied |
| APPLIED_MESSAGE_CREATE_TIME | DATE | Creation time at the source database of the last captured message applied. No information about user-enqueued messages is recorded in this column. |
| ELAPSED_DEQUEUE_TIME | NUMBER | Time elapsed (in hundredths of a second) dequeuing messages since the apply process was last started |
| ELAPSED_APPLY_TIME | NUMBER | Time elapsed (in hundredths of a second) applying messages since the apply process was last started |
| COMMIT_POSITION | RAW(64) | Commit position of the transaction. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| DEP_COMMIT_POSITION | RAW(64) | Commit position of the transaction the slave depends on. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| LAST_APPLY_POSITION | RAW(64) | For inbound servers, the position of the last message applied; for outbound servers, the position of the last message sent to the XStream client application. This column is populated only for an apply process that is functioning as an XStream outbound server or inbound server. |
| TRANSACTION_ID | VARCHAR2(128) | Transaction ID that the slave is applying. This column is populated only for an apply process that is functioning as an XStream inbound server. |

| Column | Datatype | Description |
|---|---|---|
| DEP_TRANSACTION_ID | VARCHAR2(128) | Transaction ID of the transaction the slave depends on. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| TOTAL_LCRS_RETRIED | NUMBER | Total number of LCRs retried by this server |
| LCR_RETRY_ITERATION | NUMBER | Retry iteration for this transaction by this server |
| TOTAL_TXNS_RETRIED | NUMBER | Total transactions retried by this server |
| TXN_RETRY_ITERATION | NUMBER | Retry iteration for this transaction by this server |
| TOTAL_TXNS_RECORDED | NUMBER | Total transactions recorded in error queue by this server |

**Note:**

The ELAPSED_SCHEDULE_TIME column is only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$XSTREAM_CAPTURE

V$XSTREAM_CAPTURE displays information about each capture process that sends LCRs to an XStream outbound server.

**Note:**

This view does not display information about capture processes that send LCRs to Oracle Streams apply processes. To view information about such capture processes, query the V$STREAMS_CAPTURE view.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session identifier of the capture process |
| SERIAL# | NUMBER | Session serial number of the capture process session |
| CAPTURE# | NUMBER | Capture process number. A capture process is an Oracle background process prefixed by cp |
| CAPTURE_NAME | VARCHAR2(30) | Name of the capture process |
| LOGMINER_ID | NUMBER | Session ID of the Oracle LogMiner session associated with the capture process |
| STARTUP_TIME | DATE | Time when the capture process was last started |
| STATE | VARCHAR2(551) | State of the capture process:<br>• INITIALIZING - Starting up.<br>• WAITING FOR DICTIONARY REDO - Waiting for redo log files containing the dictionary build related to the first SCN to be added to the capture process session. A capture process cannot begin to scan the redo log files until all of the log files containing the dictionary build have been added.<br>• DICTIONARY INITIALIZATION - Processing a dictionary build.<br>• MINING (PROCESSED SCN = *scn_value*) - Mining a dictionary build at the SCN *scn_value*.<br>• LOADING (step *X* of *Y*) - Processing information from a dictionary build and currently at step *X* in a process that involves *Y* steps, where *X* and *Y* are numbers.<br>• CAPTURING CHANGES - Scanning the redo log for changes that satisfy the capture process rule sets.<br>• WAITING FOR REDO - Waiting for new redo log files to be added to the capture process session. The capture process has finished processing all of the redo log files added to its session. This state is possible if there is no activity at a source database. For a downstream capture process, this state is possible if the capture process is waiting for new log files to be added to its session.<br>• EVALUATING RULE - Evaluating a change against a capture process rule set.<br>• CREATING LCR - Converting a change into an LCR.<br>• ENQUEUING MESSAGE - Enqueuing an LCR that satisfies the capture process rule sets into the capture process queue.<br>• PAUSED FOR FLOW CONTROL - Unable to enqueue LCRs either because of low memory or because propagations and outbound servers are consuming messages slower than the capture process is creating them. This state indicates flow control that is used to reduce spilling of captured LCRs when propagation or apply has fallen behind or is unavailable.<br>• WAITING FOR THE BUFFERED QUEUE TO SHRINK - Waiting for the buffered queue to change to a smaller size. The buffered queue shrinks when there is a memory limitation or when an administrator reduces its size.<br>• WAITING FOR *n* SUBSCRIBER(S) INITIALIZING - Waiting for outbound servers that receive LCRs from the capture process to start, where *n* is the number of apply processes.<br>• WAITING FOR TRANSACTION - Waiting for LogMiner to provide more transactions. |

| Column | Datatype | Description |
|---|---|---|
| | | • WAITING FOR INACTIVE DEQUEUERS - Waiting for the capture process's queue subscribers to start. The capture process stops enqueuing LCRs if there are no active subscribers to the queue.<br>• SUSPENDED FOR AUTO SPLIT/MERGE - Waiting for a merge operation to complete.<br>• SHUTTING DOWN - Stopping.<br>• ABORTING - Aborting. |
| TOTAL_PREFILTER_DISCARDED | NUMBER | Total number of prefiltered messages discarded |
| TOTAL_PREFILTER_KEPT | NUMBER | Total number of prefiltered messages kept |
| TOTAL_PREFILTER_EVALUATIONS | NUMBER | Total number of prefilter evaluations |
| TOTAL_MESSAGES_CAPTURED | NUMBER | Total number of redo entries passed by LogMiner to the capture process for detailed rule evaluation since the capture process last started. A capture process converts a redo entry into a message and performs detailed rule evaluation on the message when capture process prefiltering cannot discard the change. |
| CAPTURE_TIME | DATE | Time when the most recent message was captured |
| CAPTURE_MESSAGE_NUMBER | NUMBER | Number of the most recently captured message |
| CAPTURE_MESSAGE_CREATE_TIME | DATE | Creation time of the most recently captured message |
| TOTAL_MESSAGES_CREATED | NUMBER | Count associated with ELAPSED_LCR_TIME to calculate rate |
| TOTAL_FULL_EVALUATIONS | NUMBER | Count associated with ELAPSED_RULE_TIME to calculate rate |
| TOTAL_MESSAGES_ENQUEUED | NUMBER | Total number of messages enqueued since the capture process was last started |
| ENQUEUE_TIME | DATE | Time when the last message was enqueued |
| ENQUEUE_MESSAGE_NUMBER | NUMBER | Number of the last enqueued message |
| ENQUEUE_MESSAGE_CREATE_TIME | DATE | Creation time of the last enqueued message |
| AVAILABLE_MESSAGE_NUMBER | NUMBER | For local capture, the last redo SCN flushed to the log files. For downstream capture, the last SCN added to LogMiner through the archived redo log files. |
| AVAILABLE_MESSAGE_CREATE_TIME | DATE | For local capture, the time the SCN was written to the log file. For downstream capture, the time the most recent archived redo log file (containing the most recent SCN) was added to LogMiner. |
| ELAPSED_CAPTURE_TIME | NUMBER | Elapsed time (in hundredths of a second) scanning for changes in the redo log since the capture process was last started |

| Column | Datatype | Description |
|---|---|---|
| ELAPSED_RULE_TIME | NUMBER | Elapsed time (in hundredths of a second) evaluating rules since the capture process was last started |
| ELAPSED_ENQUEUE_TIME | NUMBER | Elapsed time (in hundredths of a second) enqueuing messages since the capture process was last started |
| ELAPSED_LCR_TIME | NUMBER | Elapsed time (in hundredths of a second) creating LCRs since the capture process was last started |
| ELAPSED_REDO_WAIT_TIME | NUMBER | Elapsed time (in hundredths of a second) spent by the capture process in the WAITING FOR REDO state |
| ELAPSED_PAUSE_TIME | NUMBER | Elapsed flow control pause time (in hundredths of a second) |
| STATE_CHANGED_TIME | DATE | Time at which the state of the capture process changed |
| SGA_USED | NUMBER | The total amount of shared memory (in bytes) currently used by the capture process out of the amount allocated (SGA_ALLOCATED) |
| SGA_ALLOCATED | NUMBER | The total amount of shared memory (in bytes) allocated from the Streams pool for the capture process |
| BYTES_OF_REDO_MINED | VARCHAR2(64) | The total amount of redo data mined (in bytes) since the capture process last started |
| SESSION_RESTART_SCN | VARCHAR2(64) | The SCN from which the capture process started mining redo data when it was last started |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**Note:**

The ELAPSED_CAPTURE_TIME, ELAPSED_RULE_TIME, ELAPSED_ENQUEUE_TIME, ELAPSED_LCR_TIME, and ELAPSED_REDO_WAIT_TIME columns are only populated if the TIMED_STATISTICS initialization parameter is set to true, or if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

**See Also:**

- "TIMED_STATISTICS"
- "STATISTICS_LEVEL"

# V$XSTREAM_MESSAGE_TRACKING

V$XSTREAM_MESSAGE_TRACKING displays information about LCRs tracked through the stream that are processed by XStream components.

You can track an LCR through a stream using one of the following methods:

- Set the message_tracking_frequency capture process parameter to 1 or another relatively low value.
- Use the DBMS_XSTREAM_ADM.SET_MESSAGE_TRACKING procedure to specify a tracking label that becomes part of each LCR generated by the current session.

  When the actions parameter in the DBMS_XSTREAM_ADM.SET_MESSAGE_TRACKING procedure is set to DBMS_XSTREAM_ADM.ACTION_MEMORY, information about the LCRs is tracked in memory, and this view is populated with information about the LCRs. Currently, DBMS_XSTREAM_ADM.ACTION_MEMORY is the only valid setting for the actions parameter in the procedure.

**Note:**

This view does not display information about messages flowing in an Oracle Streams configuration. To view information about such message streams, query the V$STREAMS_MESSAGE_TRACKING view.

This view does not display information about messages flowing in an Oracle GoldenGate configuration. To view information about such message streams, query the V$GOLDENGATE_MESSAGE_TRACKING view.

| Column | Datatype | Description |
|---|---|---|
| TRACKING_LABEL | VARCHAR2(30) | User-specified tracking label |
| TAG | RAW(30) | First 30 bytes of the tag of the LCR |
| COMPONENT_NAME | VARCHAR2(30) | Name of the component that processed the LCR |
| COMPONENT_TYPE | VARCHAR2(30) | Type of the component that processed the LCR |
| ACTION | VARCHAR2(50) | Action performed on the LCR |
| ACTION_DETAILS | VARCHAR2(100) | Details of the action |
| TIMESTAMP | TIMESTAMP(9) WITH TIME ZONE | Time when the action was performed |
| MESSAGE_CREATION_TIME | DATE | Time when the message was created |
| MESSAGE_NUMBER | NUMBER | SCN of the message |
| TRACKING_ID | RAW(16) | Globally unique OID of the LCR |
| SOURCE_DATABASE_NAME | VARCHAR2(128) | Name of the source database |

| Column | Datatype | Description |
|---|---|---|
| OBJECT_OWNER | VARCHAR2(30) | Owner of the object |
| OBJECT_NAME | VARCHAR2(30) | Name of the object |
| XID | VARCHAR2(128) | Transaction ID |
| COMMAND_TYPE | VARCHAR2(30) | Command type of the LCR |
| MESSAGE_POSITION | RAW(64) | Position of the message |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

- "V$STREAMS_MESSAGE_TRACKING"
- "V$GOLDENGATE_MESSAGE_TRACKING"
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_XSTREAM_ADM package
- *Oracle Database XStream Guide* for more information about XStream concepts

# V$XSTREAM_OUTBOUND_SERVER

V$XSTREAM_OUTBOUND_SERVER displays statistics about an outbound server. An outbound server sends LCRs to the XStream client application.

**Note:**

When the COMMITTED_DATA_ONLY column is YES in the V$XSTREAM_OUTBOUND_SERVER view, the V$STREAMS_APPLY_SERVER view provides additional information about the outbound server process, and information about the apply server background processes used by the outbound server.

| Column | Datatype | Description |
|---|---|---|
| SID | NUMBER | Session ID of the outbound server's session |
| SERIAL# | NUMBER | Serial number of the outbound server's session |
| SPID | VARCHAR2(12) | Process identification number of the operating-system process that sends LCRs to the client application |

| Column | Datatype | Description |
|---|---|---|
| SERVER_NAME | VARCHAR2(30) | Name of the outbound server |
| STARTUP_TIME | DATE | Time when the client application attached to the outbound server |
| STATE | VARCHAR2(19) | State of the outbound server<br>When the COMMITTED_DATA_ONLY column shows YES, the following states are possible:<br>• INITIALIZING - Starting up the outbound server.<br>• IDLE - Performing no work because there are no LCRs to send to the XStream client application.<br>• GET TRANSACTIONS - Receiving transactions from the outbound server's apply coordinator.<br>• SEND TRANSACTION - Sending a transaction to an XStream client application.<br>• WAIT FOR NEXT CHUNK - Waiting for the next set of LCRs for a large transaction.<br>• TRANSACTION CLEANUP - Cleaning up an applied transaction, which includes removing LCRs from the outbound server's queue.<br>• WAIT FOR CLIENT - Waiting for an XStream client application to request more LCRs.<br>When the COMMITTED_DATA_ONLY column shows NO, the following states are possible:<br>• INITIALIZING - Starting up the outbound server.<br>• INITIALIZING RULE EVALUATION CONTEXT - Initializing the context to evaluate the outbound server's rules.<br>• IDLE - Performing no work because there is no LCR to send to the XStream client application.<br>• BROWSING LCR - Browsing the outbound server's queue for next LCR.<br>• EVALUATING RULES - Evaluating an LCR against a rule set.<br>• DEQUEUING LCR - Dequeuing an LCR from the outbound server's queue.<br>• SENDING LCR - Sending an LCR to an XStream client application.<br>• WAITING FOR CAPTURE TO TERMINATE - Waiting for the capture process to become disabled.<br>• SUSPENDED DUE TO A DROPPED SUBSCRIBER - Suspended because a connected subscriber was dropped. For example, a subscriber can be dropped during a split or merge operation.<br>• SUSPENDED FOR AUTO SPLIT/MERGE - Suspended because an automatic split or merge operation is being performed.<br>• WAITING ON EMPTY QUEUE - Waiting for more LCRs from the capture process.<br>• WAITING FOR CLIENT - Waiting for the XStream client application to request more LCRs.<br>• WAITING FOR CAPTURE TO INITIALIZE - Waiting for the capture process to finish the data dictionary build.<br>• WAITING TO ATTACH TO CAPTURE - Waiting for the outbound server to attach to the capture process.<br>When a state refers to a capture process, it is the capture process that captures changes for the outbound server. When a state refers to a propagation, it is the outbound server that sends LCRs to the XStream client application. |

| Column | Datatype | Description |
|---|---|---|
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction currently being processed. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction currently being processed. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction currently being processed. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |
| COMMITSCN | NUMBER | Commit SCN of the transaction currently being processed. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |
| TOTAL_TRANSACTIONS_SENT | NUMBER | Total number of transactions sent by the outbound server to the XStream client application since the last time the client application attached to the outbound server. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |
| MESSAGE_SEQUENCE | NUMBER | Number of the current LCR being processed by the outbound server. This value is reset to 1 at the beginning of each transaction. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |
| TOTAL_MESSAGES_SENT | NUMBER | Total number of LCRs sent by the outbound server to the XStream client application since the last time the client application attached to the outbound server |
| SEND_TIME | DATE | Time the last LCR was sent by the outbound server to the XStream client application |
| LAST_SENT_MESSAGE_NUMBER | NUMBER | Message number of the last LCR sent by the outbound server to the XStream client application |
| LAST_SENT_MESSAGE_CREATE_TIME | DATE | Creation time at the source database of the last LCR sent by the outbound server to the client application |
| ELAPSED_SEND_TIME | NUMBER | Time elapsed (in hundredths of a second) sending LCRs to the XStream client application since the last time the client application attached to the outbound server |
| COMMIT_POSITION | RAW(64) | Commit position of the transaction currently being processed. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |

| Column | Datatype | Description |
|---|---|---|
| LAST_SENT_POSITION | RAW(64) | Position of the last LCR sent to the XStream client application. This column is populated only if the COMMITTED_DATA_ONLY column shows YES. When the COMMITTED_DATA_ONLY column shows NO, this column is NULL. |
| BYTES_SENT | NUMBER | Total number of bytes sent by the outbound server to the XStream client application since the last time the client application attached to the outbound server |
| COMMITTED_DATA_ONLY | CHAR(3) | YES if the outbound server can send only LCRs in committed transactions to the XStream client application. A committed transaction is an assembled, noninterleaving transaction with no rollbacks.<br>NO if the outbound server can send LCRs in transactions that have not yet committed to the XStream client application. This mode is for internal Oracle use only. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$STREAMS_APPLY_SERVER"

# V$XSTREAM_TABLE_STATS

V$XSTREAM_TABLE_STATS shows the statistics for all the tables processed by each apply server for the current session.

| Column | Datatype | Description |
|---|---|---|
| APPLY_NAME | VARCHAR2(30) | Name of XStream Out or In process |
| SERVER_ID | NUMBER | Parallel apply server slave ID. If the server ID is 0, then this is an aggregate of statistics for servers that may have been automatically shutdown due to session inactivity. |
| SOURCE_TABLE_OWNER | VARCHAR2(30) | Source owner of the captured or replicated table |
| SOURCE_TABLE_NAME | VARCHAR2(30) | Source name of the captured or replicated table |
| DESTINATION_TABLE_OWNER | VARCHAR2(30) | Target owner of the captured or replicated table |
| DESTINATION_TABLE_NAME | VARCHAR2(30) | Target name of the captured or replicated table |
| LAST_UPDATE | DATE | Time of last update |

| Column | Datatype | Description |
|---|---|---|
| TOTAL_INSERTS | NUMBER | Number of insert operations on this table processed by this apply server for the current session |
| TOTAL_UPDATES | NUMBER | Number of update operations on this table processed by this apply server for the current session |
| TOTAL_DELETES | NUMBER | Number of delete operations on this table processed by this apply server for the current session |
| INSERT_COLLISIONS | NUMBER | Number of insert collisions on this table encountered by this apply server for the current session |
| UPDATE_COLLISIONS | NUMBER | Number of update collisions on this table encountered by this apply server for the current session |
| DELETE_COLLISIONS | NUMBER | Number of delete collisions on this table encountered by this apply server for the current session |
| REPERROR_RECORDS | NUMBER | Number of change records that were recorded on this table by this apply server for the current session |
| REPERROR_IGNORES | NUMBER | Number of ignored change records on this table by this apply server for the current session |
| WAIT_DEPENDENCIES | NUMBER | Number of waits for this table due to dependency for the current session |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |
| CDR_INSERT_ROW_EXISTS | NUMBER | Number of conflicts where an insert gets an ORA-00001 error |
| CDR_UPDATE_ROW_EXISTS | NUMBER | Number of conflicts where an update gets an ORA-26786 error |
| CDR_UPDATE_ROW_MISSING | NUMBER | Number of conflicts where an update gets an ORA-26787 error |
| CDR_DELETE_ROW_EXISTS | NUMBER | Number of conflicts where a delete gets an ORA-26786 error |
| CDR_DELETE_ROW_MISSING | NUMBER | Number of conflicts where a delete gets an ORA-26787 error |
| CDR_SUCCESSFUL_RESOLUTIONS | NUMBER | Number of successfully resolved conflicts |
| CDR_FAILED_RESOLUTIONS | NUMBER | Number of conflicts that could not be resolved due to an error during resolution |
| LOB_OPERATIONS[1] | NUMBER | The number of LOB updates (LOB writes, LOB trims, and LOB erases) applied by the inbound server. |

[1] This column is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

**See Also:**

This note was originally conditionalized to DB122.

*Oracle Database XStream Guide* for more information about XStream conflict detection and resolution.

# V$XSTREAM_TRANSACTION

V$XSTREAM_TRANSACTION displays information about transactions that are being processed by capture processes, outbound servers, and inbound servers. This view can identify long running transactions and to display how many LCRs are being processed in each transaction. This view only contains information about captured LCRs. It does not contain information about user-enqueued LCRs or user messages.

This view only shows information about LCRs that are being processed because they satisfied the rule sets for the component at the time of the query. For capture processes, this view only shows information about changes in transactions that the capture process has converted into LCRs. It does not show information about all the active transactions present in the redo log.

For outbound servers, this view only shows information about LCRs that the outbound server has dequeued. It does not show information about LCRs in the outbound server's queue. For outbound servers, information about a transaction remains in the view until the transaction is sent to the XStream client application.

For inbound servers, information about a transaction remains in the view until the transaction commits or until the entire transaction is rolled back.

**Note:**

This view does not display information about Oracle Streams transactions. To view information about Oracle Streams transactions, query the V$STREAMS_TRANSACTION view.

| Column | Datatype | Description |
|---|---|---|
| COMPONENT_NAME | VARCHAR2(30) | Name of the component |
| COMPONENT_TYPE | VARCHAR2(20) | Type of component<br>• CAPTURE - Capture process<br>• APPLY - Apply reader subcomponent in an outbound server or inbound server<br>• PROPAGATION_SENDER - Propagation sender that sends LCRs from a capture process to an outbound server |
| XIDUSN | NUMBER | Transaction ID undo segment number of the transaction |
| XIDSLT | NUMBER | Transaction ID slot number of the transaction |
| XIDSQN | NUMBER | Transaction ID sequence number of the transaction |
| CUMULATIVE_MESSAGE_COUNT | NUMBER | Number of LCRs processed in the transaction. If a component is restarted while the transaction is being processed, then this |

| Column | Datatype | Description |
|---|---|---|
| | | column shows the number of LCRs processed in the transaction since the component was started. |
| TOTAL_MESSAGE_COUNT | NUMBER | Total number of LCRs processed in the transaction by an outbound server or inbound server. This column does not pertain to capture processes. |
| FIRST_MESSAGE_TIME | DATE | Time stamp of the first LCR processed in the transaction. If a capture process is restarted while the transaction is being processed, then this column shows the time stamp of the first LCR processed after the capture process was started. |
| FIRST_MESSAGE_NUMBER | NUMBER | SCN of the first message in the transaction. If a capture process is restarted while the transaction is being processed, then this column shows the SCN of the first message processed after the capture process was started. |
| LAST_MESSAGE_TIME | DATE | Time stamp of the last LCR processed in the transaction |
| LAST_MESSAGE_NUMBER | NUMBER | SCN of the most recent message encountered for the transaction |
| FIRST_MESSAGE_POSITION | RAW(64) | Position of the first message seen by an XStream inbound server<br>This column is populated only for an apply process that is functioning as an XStream inbound server. |
| LAST_MESSAGE_POSITION | RAW(64) | Position of the last message seen by an XStream inbound server. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| TRANSACTION_ID | VARCHAR2(128) | Transaction ID for an XStream inbound server. This column is populated only for an apply process that is functioning as an XStream inbound server. |
| CON_ID | NUMBER | The ID of the container to which the data pertains. Possible values include:<br>• 0: This value is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.<br>• 1: This value is used for rows containing data that pertain to only the root<br>• *n*: Where *n* is the applicable container ID for the rows containing data |

**See Also:**

"V$STREAMS_TRANSACTION"

# Part IV

## Appendixes

This part includes the following *Oracle Database Reference* manual appendixes:

- Database Limits
- SQL Scripts
- Oracle Wait Events
- Oracle Enqueue Names
- Statistics Descriptions
- Background Processes

# A

# Database Limits

This appendix lists the limits of values associated with database functions and objects. Limits exist on several levels in the database. There is usually a hard-coded limit in the database that cannot be exceeded. This value may be further restricted for any given operating system.

Database limits are divided into four categories:

- Datatype Limits
- Physical Database Limits
- Logical Database Limits
- Process and Runtime Limits

**See Also:**

Your operating system-specific Oracle documentation

## Datatype Limits

| Datatypes | Limit | Comments |
|---|---|---|
| BFILE | Maximum size: 4 GB<br>Maximum size of a file name: 255 characters<br>Maximum size of a directory name: 30 characters<br>Maximum number of open BFILEs: see Comments | The maximum number of BFILEs is limited by the value of the SESSION_MAX_OPEN_FILES initialization parameter, which is itself limited by the maximum number of open files the operating system will allow. |
| BLOB | Maximum size: (4 GB - 1) * DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB) | The number of LOB columns per table is limited only by the maximum number of columns per table (that is, 1000). |
| CHAR | Maximum size: 2000 bytes | None |
| CHAR VARYING | Maximum size: 4000 bytes | None |
| CLOB | Maximum size: (4 GB - 1) * DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB) | The number of LOB columns per table is limited only by the maximum number of columns per table (that is, 1000[1]). |
| Literals (characters or numbers in SQL or PL/SQL) | Maximum size: 4000 characters | None |

| Datatypes | Limit | Comments |
|---|---|---|
| LONG | Maximum size: 2 GB - 1 | Only one LONG column is allowed per table. |
| NCHAR | Maximum size: 2000 bytes | None |
| NCHAR VARYING | Maximum size: 4000 bytes, or 32767 bytes if the MAX_STRING_SIZE initialization parameter is set to EXTENDED<br>**See Also:** "MAX_STRING_SIZE" initialization parameter for additional details | None |
| NCLOB | Maximum size: (4 GB - 1) * DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB) | The number of LOB columns per table is limited only by the maximum number of columns per table (that is, 1000[1]). |
| NUMBER | 999...(38 9's) x$10^{125}$ maximum value | Can be represented to full 38-digit precision (the mantissa) |
| | -999...(38 9's) x$10^{125}$ minimum value | Can be represented to full 38-digit precision (the mantissa) |
| Precision | 38 significant digits | None |
| RAW | Maximum size: 2000 bytes, or 32767 bytes if the MAX_STRING_SIZE initialization parameter is set to EXTENDED<br>**See Also:** "MAX_STRING_SIZE" initialization parameter for additional details | None |
| VARCHAR | Maximum size: 4000 bytes | None |
| VARCHAR2 | Maximum size: 4000 bytes, or 32767 bytes if the MAX_STRING_SIZE initialization parameter is set to EXTENDED<br>**See Also:** "MAX_STRING_SIZE" initialization parameter for additional details | None |

[1] The absolute maximum number of columns in a table is 1000. However, when you create an object table (or a relational table with columns of object, nested table, varray, or REF type), Oracle maps the columns of the user-defined types to relational columns, creating in effect hidden columns that count toward the 1000-column limit. For details on how Oracle calculates the total number of columns in such a table, refer to *Oracle Database Administrator's Guide*.

**See Also:**

- *Oracle Database SQL Language Reference* for more information about datatypes
- *Oracle Database SQL Language Reference* for more information on naming database objects

## Physical Database Limits

| Item | Type of Limit | Limit Value |
|---|---|---|
| Database Block Size | Minimum | 2048 bytes; must be a multiple of operating system physical block size |
| Database Block Size | Maximum | Operating system dependent; never more than 32 KB |
| Database Blocks | Minimum in initial extent of a segment | 2 blocks |
| Database Blocks | Maximum per datafile | Platform dependent; typically $2^{22}$ - 1 blocks |
| Controlfiles | Number of control files | 1 minimum; 2 or more (on separate devices) strongly recommended |
| Controlfiles | Size of a control file | Maximum of 201031680 logical blocks |
| Database files | Maximum per tablespace | Operating system dependent; usually 1022 |
| Database files | Maximum per database | 65533<br>May be less on some operating systems<br>Limited also by size of database blocks and by the DB_FILES initialization parameter for a particular instance |
| Database extents | Maximum per dictionary managed tablespace | 4 GB * physical block size (with K/M modifier); 4 GB (without K/M modifier) |
| Database extents | Maximum per locally managed (uniform) tablespace | 2 GB * physical block size (with K/M modifier); 2 GB (without K/M modifier) |
| Database file size | Maximum | Operating system dependent. Limited by maximum operating system file size. See the **Bigfile Tablespaces** and **Smallfile (traditional) Tablespaces** rows for more information about the maximum database file size in these types of tablespaces. |
| MAXEXTENTS | Default value | Derived from tablespace default storage or DB_BLOCK_SIZE initialization parameter |
| MAXEXTENTS | Maximum | Unlimited |
| Redo Log Files | Maximum number of logfiles | Limited by value of MAXLOGFILES parameter in the CREATE DATABASE statement<br>Control file can be resized to allow more entries; ultimately an operating system limit |

| Item | Type of Limit | Limit Value |
|---|---|---|
| Redo Log Files | Maximum number of logfiles per group | Unlimited |
| Redo Log File Size | Minimum size | 4 MB |
| Redo Log File Size | Maximum Size | Operating system limit; typically 2 GB |
| Tablespaces | Maximum number per database | 64 K<br>Number of tablespaces cannot exceed the number of database files because each tablespace must include at least one file |
| Bigfile Tablespaces | Number of blocks | A bigfile tablespace contains only one datafile or tempfile, which can contain up to approximately 4 billion ( $2^{32}$ ) blocks. The maximum size of the single datafile or tempfile is 128 terabytes (TB) for a tablespace with 32 K blocks and 32 TB for a tablespace with 8 K blocks. |
| Smallfile (traditional) Tablespaces | Number of blocks | A smallfile tablespace is a traditional Oracle tablespace, which can contain 1022 datafiles or tempfiles, each of which can contain up to approximately 4 million ($2^{22}$) blocks. |
| External Tables file | Maximum size | Dependent on the operating system.<br>An external table can be composed of multiple files. |

## Logical Database Limits

| Item | Type of Limit | Limit Value |
|---|---|---|
| Indexes | Maximum per table | Unlimited |
| Indexes | Total size of indexed column | 75% of the database block size minus some overhead |
| Columns | Per table | 1000 columns maximum |
| Columns | Per index (or clustered index) | 32 columns maximum |
| Columns | Per bitmapped index | 30 columns maximum |
| Constraints | Maximum per column | Unlimited |
| Subqueries | Maximum levels of subqueries in a SQL statement | Unlimited in the FROM clause of the top-level query<br>255 subqueries in the WHERE clause |
| Partitions | Maximum length of linear partitioning key | 4 KB - overhead |
| Partitions | Maximum number of columns in partition key | 16 columns |
| Partitions | Maximum number of partitions allowed per table or index | 1024K - 1 |

| Item | Type of Limit | Limit Value |
|---|---|---|
| Subpartitions | Maximum number of subpartitions in a composite partitioned table | 1024K - 1 |
| Rows | Maximum number per table | Unlimited |
| System Change Numbers (SCNs) | Maximum | 281,474,976,710,656, which is 281 trillion SCNs |
| Stored Packages | Maximum size | Approximately `6,000,000` lines of code.<br>**See Also:** *Oracle Database PL/SQL Language Reference* for details |
| Trigger Cascade Limit | Maximum value | Operating system-dependent, typically `32` |
| Users and Roles | Maximum | `2,147,483,638` |
| Tables | Maximum per clustered table | 32 tables |
| Tables | Maximum per database | Unlimited |

**Note:**

The limit on how long a SQL statement can be depends on many factors, including database configuration, disk space, and memory

**Note:**

When an object instance exists in memory, there is no fixed limit on the number of attributes in the object. But the maximum total amount of memory consumed by an object instance is 4 GB.When an object instance is inserted into a table, the attributes are exploded into separate columns in the table, and the Oracle 1000-column limit applies.

## Process and Runtime Limits

| Item | Type of Limit | Limit Value |
|---|---|---|
| Instances per database | Maximum number of cluster database instances per database | Operating system-dependent |
| Locks | Row-level | Unlimited |
| Locks | Distributed Lock Manager | Operating system dependent |
| SGA size | Maximum value | Operating system-dependent; typically 2 to 4 GB for 32-bit operating systems, and > 4 GB for 64-bit operating systems |
| Advanced Queuing Processes | Maximum per instance | `10` |
| Job Queue Processes | Maximum per instance | `1000` |

| Item | Type of Limit | Limit Value |
|---|---|---|
| I/O Slave Processes | Maximum per background process (DBWR, LGWR, and so on) | 15 |
| I/O Slave Processes | Maximum per Backup session | 15 |
| Sessions | Maximum per instance | $2^{16}$; limited by the `PROCESSES` and `SESSIONS` initialization parameters. $2^{16}$ is 65536. |
| Global Cache Service Processes | Maximum per instance | `10` |
| Shared Servers | Maximum per instance | Unlimited within constraints set by the `PROCESSES` and `SESSIONS` initialization parameters, for instance |
| Dispatchers | Maximum per instance | Unlimited within constraints set by `PROCESSES` and `SESSIONS` initialization parameters, for instance |
| Parallel Execution Slaves | Maximum per instance | Unlimited within constraints set by `PROCESSES` and `SESSIONS` initialization parameters, for instance |
| Backup Sessions | Maximum per instance | Unlimited within constraints set by `PROCESSES` and `SESSIONS` initialization parameters, for instance |
| Services | Maximum per instance | 1024 |

# B

# SQL Scripts

This appendix describes SQL scripts that are required for optimal operation of the Oracle Database.

The SQL scripts are described in the following sections:

- Creating the Data Dictionary
- Creating Additional Data Dictionary Structures
- The "NO" Scripts
- Upgrade Scripts
- Java Scripts

**Note:**

Comments within the SQL scripts themselves contain more detailed information and examples.

**Note:**

To run scripts in a pluggable database (PDB), connect to the PDB before running the script. For more information about running scripts in a multitenant container database (CDB), see *Oracle Database Administrator's Guide*.

## Creating the Data Dictionary

When you use the Database Configuration Assistant to create a database, Oracle automatically creates the data dictionary. Thereafter, whenever the database is in operation, Oracle updates the data dictionary in response to every DDL statement.

The data dictionary base tables are the first objects created in any Oracle database. They are created in the `SYSTEM` tablespace and must remain there. The data dictionary base tables store information about all user-defined objects in the database.

Table B-1 lists required scripts, which are run automatically when you create a database using the Database Configuration Assistant. They are described here because you might need to run them if you create a database manually. To run these scripts, you must be connected to Oracle as a user with `SYSDBA` privileges.

***Table B-1 Creating the Data Dictionary Scripts***

| Script Name | Needed For | Description |
|---|---|---|
| `catalog.sql` | All databases | Creates the data dictionary and public synonyms for many of its views<br>Grants `PUBLIC` access to the synonyms<br>**See Also:** *Oracle Database Upgrade Guide* |
| `catproc.sql` | All databases | Runs all scripts required for, or used with, PL/SQL<br>**See Also:** *Oracle Database Upgrade Guide* |
| `catclust.sql` | Oracle Real Application Clusters | Creates Oracle Real Application Clusters data dictionary views<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* |

**See Also:**

- Your operating system-specific Oracle documentation for the exact names and locations of these scripts on your operating system
- *Oracle Database Administrator's Guide* for more information about creating a database
- Your release notes and *Oracle Database Upgrade Guide* to learn when it is necessary to run these scripts
- *Oracle Real Application Clusters Administration and Deployment Guide* for more information about creating Oracle Real Application Cluster views

## Creating Additional Data Dictionary Structures

Oracle Database supplies other scripts that create additional structures you can use in managing your database and creating database applications. These scripts are listed in Table B-2.

**See Also:**

Your operating system-specific Oracle documentation for the exact names and locations of these scripts on your operating system

***Table B-2 Creating Additional Data Dictionary Structures***

| Script Name | Needed For | Run By | Description |
|---|---|---|---|
| `catblock.sql` | Performance management | `SYS` | Creates views that can dynamically display lock dependency graphs<br>**See Also:** *Oracle Database Administrator's Guide* |
| `caths.sql` | Heterogeneous Services | `SYS` | Installs packages for administering heterogeneous services |

| Script Name | Needed For | Run By | Description |
|---|---|---|---|
| | | | **See Also:** *Oracle Database Gateway for APPC Installation and Configuration Guide for Microsoft Windows* |
| `catio.sql` | Performance management | `SYS` | Allows I/O to be traced on a table-by-table basis |
| `catqueue.sql` | Advanced Queuing | | Creates the dictionary objects required for Advanced Queuing |
| `catrep.sql` | Oracle Replication | `SYS` | Runs all SQL scripts for enabling database replication |
| `catwrr.sql` | Database Replay | `SYS` | Master script that creates the entire schema related to Database Replay - calls the create schema scripts for Workload Capture, Database Replay, Workload Replay, and Workload Intelligence |
| `catwrrwitb.sql` | Database Replay | `SYS` | Creates the schema for Workload Intelligence |
| `dbmsiotc.sql` | Storage management | Any user | Analyzes chained rows in index-organized tables |
| `dbmspool.sql` | Performance management | `SYS` or `SYSDBA` | Enables DBA to lock PL/SQL packages, SQL statements, and triggers into the shared pool<br>**See Also:** *Oracle Database PL/SQL Packages and Types Reference* |
| `userlock.sql` | Concurrency control | `SYS` or `SYSDBA` | Provides a facility for user-named locks that can be used in a local or clustered environment to aid in sequencing application actions |
| `utlbstat.sql` and `utlestat.sql` | Performance monitoring | `SYS` | Respectively start and stop collecting performance tuning statistics |
| `utlchn1.sql` | Storage management | Any user | For use with the Oracle Database. Creates tables for storing the output of the `ANALYZE` command with the `CHAINED ROWS` option. Can handle both physical and logical rowids.<br>**See Also:** *Oracle Database Administrator's Guide* |
| `utlconst.sql` | Year 2000 compliance | Any user | Provides functions to validate that `CHECK` constraints on date columns are year 2000 compliant |
| `utldtree.sql` | Metadata management | Any user | Creates tables and views that show dependencies between objects<br>**See Also:** *Oracle Database SecureFiles and Large Objects Developer's Guide* |
| `utlexpt1.sql` | Constraints | Any user | For use with the Oracle Database. Creates the default table (`EXCEPTIONS`) for storing exceptions from enabling |

| Script Name | Needed For | Run By | Description |
|---|---|---|---|
| | | | constraints. Can handle both physical and logical rowids.<br>**See Also:** *Oracle Database Administrator's Guide* |
| utlip.sql | PL/SQL | SYS | Used primarily for upgrade and downgrade operations. It invalidates all existing PL/SQL modules by altering certain dictionary tables so that subsequent recompilations will occur in the format required by the database. It also reloads the packages STANDARD and DBMS_STANDARD, which are necessary for any PL/SQL compilations. |
| utlirp.sql | PL/SQL | SYS | Used to change from 32-bit to 64-bit word size or vice versa. This script recompiles existing PL/SQL modules in the format required by the new database. It first alters some data dictionary tables. Then it reloads the packages STANDARD and DBMS_STANDARD, which are necessary for using PL/SQL. Finally, it triggers a recompilation of all PL/SQL modules, such as packages, procedures, and types. |
| utllockt.sql | Performance monitoring | SYS or SYSDBA | Displays a lock wait-for graph, in tree structure format<br>**See Also:** *Oracle Database Administrator's Guide* |
| utlpwdmg.sql | Security | SYS or SYSDBA | Creates PL/SQL functions for default password complexity verification. Sets the default password profile parameters and enables password management features.<br>**See Also:** *Oracle Database Security Guide* |
| utlrp.sql | PL/SQL | SYS | Recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, and types.<br>**See Also:** *Oracle Database Development Guide* |
| utlsampl.sql | Examples | SYS or any user with DBA role | Creates sample tables, such as emp and dept, and users, such as scott |
| utlscln.sql | Oracle Replication | Any user | Copies a snapshot schema from another snapshot site |
| utltkprf.sql | Performance management | SYS | Creates the TKPROFER role to allow the TKPROF profiling utility to be run by non-DBA users |
| utlvalid.sql | Partitioned tables | Any user | Creates tables required for storing output of ANALYZE TABLE ...VALIDATE STRUCTURE of a partitioned table |

| Script Name | Needed For | Run By | Description |
|---|---|---|---|
| utlxplan.sql | Performance management | Any user | Creates the table PLAN_TABLE, which holds output from the EXPLAIN PLAN statement<br>See Also: *Oracle Database Data Warehousing Guide* |

# The "NO" Scripts

The scripts in Table B-3 are used to remove dictionary information for various optional services or components.

***Table B-3 The NO Scripts***

| Script Name | Needed For | Run By | Description |
|---|---|---|---|
| catnoadt.sql | Objects | SYS | Drops views and synonyms on dictionary metadata that relate to object types |
| catnoaud.sql | Security | SYS | Drops views and synonyms on auditing metadata |
| catnoclust.sql | Real Application Clusters | SYS | Drops views that are specific to Oracle Real Application Clusters (Oracle RAC) |
| catnodrdaas.sql | DRDA Application Server | SYS | Drops all DRDA Application Server objects from the database |
| catnogwm.sql | Global Data Services | SYS | Drops users, roles, and other objects created for Global Data Services. |
| catnohs.sql | Heterogeneous Services | SYS | Removes Heterogeneous Services dictionary metadata |
| catnojav.sql | Java | SYS | Drops the RDBMS Java classes and system triggers created by the catjava.sql script. |
| catnoprt.sql | Partitioning | SYS | Drops views and synonyms on dictionary metadata that relate to partitioned tables and indexes |
| catnosvm.sql | Server Manager | SYS | Removes Oracle7 Server Manager views and synonyms |
| catnowrr.sql | Database Replay | SYS | Master script that drops the entire schema related to Database Replay - calls the drop schema scripts for Workload Capture, Database Replay, Workload Replay, and Workload Intelligence |
| catnowrrc.sql | Database Replay | SYS | Drops the schema related to Workload Capture |
| catnowrrp.sql | Database Replay | SYS | Drops the schema related to Workload Replay |

| Script Name | Needed For | Run By | Description |
|---|---|---|---|
| catnowrrwitb.sql | Database Replay | SYS | Drops the schema related to Workload Intelligence |
| catnsnmp.sql | Distributed management | SYS | Drops the DBSNMP user and SNMPAGENT role |

# Upgrade Scripts

Upgrade scripts are used when upgrading to another release of Oracle.

**See Also:**

*Oracle Database Upgrade Guide* for information about upgrade scripts

# Java Scripts

The scripts in Table B-4 are useful only if the JServer option is installed.

***Table B-4 Java Scripts***

| Script Name | Description |
|---|---|
| initjvm.sql | Initializes JServer by installing core Java class libraries and Oracle-specific Java classes |
| rmjvm.sql | Removes all elements of the JServer<br>**See Also:** *Oracle Database Java Developer's Guide* |
| catjava.sql | Installs Java-related packages and classes |

# C

# Oracle Wait Events

This appendix contains the following topics:


- Classes of Wait Events
- Descriptions of Common Wait Event Parameters
- Descriptions of Wait Events


Information about wait events is displayed in three dynamic performance views:

- `V$SESSION_WAIT` displays the events for which sessions have just completed waiting or are currently waiting.
- `V$SYSTEM_EVENT` displays the total number of times all the sessions have waited for the events in that view.
- `V$SESSION_EVENT` is similar to `V$SYSTEM_EVENT`, but displays all waits for each session.

> **See Also:**
>
> "V$SESSION_EVENT", "V$SESSION_WAIT", and "V$SYSTEM_EVENT"

Many of these wait events are tied to the internal implementation of Oracle and therefore are subject to change or deletion without notice. Application developers should be aware of this and write their code to tolerate missing or extra wait events.

The following SQL statement displays an alphabetical list of all Oracle wait events and the wait class to which they belong:

```
SQL> SELECT name, wait_class FROM V$EVENT_NAME ORDER BY name;
```

## Classes of Wait Events

Every wait event belongs to a class of wait event. The following list describes each of the wait classes.

### Administrative

Waits resulting from DBA commands that cause users to wait (for example, an index rebuild)

### Application

Waits resulting from user application code (for example, lock waits caused by row level locking or explicit lock commands)

#### Cluster

Waits related to Oracle Real Application Clusters resources (for example, global cache resources such as 'gc cr block busy')

#### Commit

This wait class only comprises one wait event - wait for redo log write confirmation after a commit (that is, 'log file sync')

#### Concurrency

Waits for internal database resources (for example, latches)

#### Configuration

Waits caused by inadequate configuration of database or instance resources (for example, undersized log file sizes, shared pool size)

#### Idle

Waits that signify the session is inactive, waiting for work (for example, 'SQL*Net message from client')

#### Network

Waits related to network messaging (for example, 'SQL*Net more data to dblink')

#### Other

Waits which should not typically occur on a system (for example, 'wait for EMON to spawn')

#### Queue

Contains events that signify delays in obtaining additional data in a pipelined environment. The time spent on these wait events indicates inefficiency or other problems in the pipeline. It affects features such as Oracle Streams, parallel queries, or `DBMS_PIPE` PL/SQL packages.

#### Scheduler

Resource Manager related waits (for example, 'resmgr: become active')

#### System I/O

Waits for background process I/O (for example, DBWR wait for 'db file parallel write')

#### User I/O

Waits for user I/O (for example 'db file sequential read')

## Descriptions of Common Wait Event Parameters

This section describes some of the common wait event parameters.

### block#

This is the block number of the block for which Oracle needs to wait. The block number is relative to the start of the file. To find the object to which this block belongs, issue the following SQL statement:

```
select segment_name, segment_type, owner, tablespace_name
from dba_extents
where file_id = file#
     and block#
     between block_id and block_id + blocks - 1;
```

### blocks

The number of blocks that is being either read from or written to the file. The block size depends on the file type:

- Database files have a block size of `DB_BLOCK_SIZE`
- Logfiles and control files have a block size that is equivalent to the physical block size of the platform

### break?

If the value for this parameter equals 0, a reset was sent to the client. A nonzero value indicates that a break was sent to the client.

### class

The class of the block describes how the contents of the block are used. For example, class 1 represents data block, and class 4 represents segment header.

### dba

The initials "dba" represents the data block address, which consists of a file number and a block number.

### driver id

The address of the disconnect function of the driver that is currently being used.

### file#

The following query returns the name of the database file:

```
select *
from v$datafile
where file# = file#;
```

### id1

The first identifier (*id1*) of the enqueue or global lock takes its value from P2 or P2RAW. The meaning of the identifier depends on the name (P1).

### id2

The second identifier (*id2*) of the enqueue or global lock takes its value from P3 or P3RAW. The meaning of the identifier depends on the name (P1).

### le

The relative index number into V$GC_ELEMENT.

### mode

The *mode* is usually stored in the low order bytes of P1 or P1RAW and indicates the mode of the enqueue or global lock request. This parameter has one of the following values:

***Table C-1 Lock Mode Values***

| Mode Value | Description |
|---|---|
| 1 | Null mode |
| 2 | Sub-Share |
| 3 | Sub-Exclusive |
| 4 | Share |
| 5 | Share/Sub-Exclusive |
| 6 | Exclusive |

Use the following SQL statement to retrieve the name of the lock and the mode of the lock request:

```
select chr(bitand(p1,-16777216)/16777215)||
       chr(bitand(p1, 16711680)/65535) "Lock",
       bitand(p1, 65535) "Mode"
from v$session_wait
where event = 'DFS enqueue lock acquisition';
```

### name and type

The name or "type" of the enqueue or global lock can be determined by looking at the two high order bytes of P1 or P1RAW. The name is always two characters. Use the following SQL statement to retrieve the lock name.

```
select chr(bitand(p1,-16777216)/16777215)||
      chr(bitand(p1,16711680)/65535) "Lock"
from v$session_wait
where event = 'DFS enqueue lock acquisition';
```

### namespace

The name of the object namespace as it is displayed in V$DB_OBJECT_CACHE view.

### requests

The number of I/Os that are "requested." This differs from the number of blocks in that one request could potentially contain multiple blocks.

### session#

The number of the inactive session. Use the following SQL statement to find more information about the session:

```
select *
from v$session
where sid = session#;
```

**waited**

This is the total amount of time the session has waited for this session to terminate.

# Descriptions of Wait Events

This section provides detailed descriptions for those wait events of greatest interest. Where appropriate, pointers are provided to further information elsewhere in Oracle Database documentation. For a complete listing of wait events, in alphabetical order, you can issue the following SQL statement:

```
SQL> SELECT name FROM V$EVENT_NAME ORDER BY name;
```

## alter system set dispatcher

A session has issued a statement `ALTER SYSTEM SET DISPATCHER = string` and is waiting for the dispatchers to get started.

**Wait Time:** The session will wait 1 / 100 of a second and verify if the new dispatchers have started, else the session will wait again

| Parameter | Description |
|---|---|
| *waited* | Number of times that the session has waited 1 / 100 of a second |

## ARCH Remote Write

Used to track the time (in centi-seconds) that ARC*n* background processes spend blocked waiting for network write operations to complete.

## ASYNC Remote Write

Used to track the time (in centi-seconds) for asynchronous streaming RFSWRITE operations. This includes stall reaps and streaming network submission time. This time is accumulated by TT*nn* (Redo Transport Slave) background processes.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about using RFS to manage standby redo logs.

## batched allocate scn lock request

A session is waiting on another process to allocate a system change number (SCN). If the foreground timed out waiting on a process to get the SCN, the foreground will get the SCN.

**Wait Time:** The wait time is 1 second on the assumption that an SCN allocation should normally need much less than that

**Parameters:** None

## BFILE check if exists

The session waits to check if an external large object (LOB) exists.

**Wait Time:** The total elapsed time for the **exists** call

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## BFILE check if open

The session waits for an external large object (LOB) to open.

**Wait Time:** The total elapsed time for the **isopen** call

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## BFILE closure

The session waits for an external large object (LOB) to close.

**Wait Time:** The total elapsed time for the **close** call

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## BFILE get length

The session waits on a call to check the size of an external large object (LOB).

**Wait Time:** The total elapsed time for the call to check the LOB size

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## BFILE get name object

The session waits on a call to find or generate the external name of a external large object.

**Wait Time:** The total elapsed time for **make external file name** to complete

| Parameter | Description |
|---|---|
| *session#* | See "session#" |

| Parameter | Description |
|---|---|
| *waited* | See "waited" |

## BFILE get path object

The session is waiting on a call to find or generate the external path name of an external large object (LOB).

**Wait Time:** The total elapsed time for **make external path** to complete

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## BFILE internal seek

The session waits for a positioning call within the external large object (LOB) to complete.

**Wait Time:** The total elapsed time for the **seek** to complete

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## BFILE open

The session waits for an external large object (LOB) to open.

**Wait Time:** The total elapsed time for the **isopen** call

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## BFILE read

The session waits for a read from a external large object (LOB) to complete.

**Wait Time:** The total elapsed time for the **read** to complete

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## broadcast mesg queue transition

Processes enter "wait for broadcast mesg queue transition" when cleaning up a publisher channel handle to a RELIABLE broadcast channel. The publisher is responsible for moving the message to the free queue, but it cannot do so until the

message is in the done queue. If the message is still not in the done queue, process enters this wait. This wait event will most likely appear when an Oracle process is about to exit normally, or when PMON cleans up a dead process.

**Wait Time:** Varies

| Parameter | Description |
|---|---|
| *channel handle* | publisher channel handle pointer |
| *message* | broadcast message pointer |
| *location* | A number indicating the function in KSR where the process is waiting |

## broadcast mesg recovery queue transition

Processes enter "wait for broadcast mesg recovery queue transition" when cleaning up a publisher channel handle to a RELIABLE broadcast channel. The broadcasted message is in the recovery queue of another channel handle (for example, ch2). Process enters this wait, if the message is yet to be removed from the recovery queue of the ch2 channel handle. This wait event will most likely appear when an Oracle process is about to exit normally, or when PMON cleans up a dead process.

**Wait Time:** Varies

| Parameter | Description |
|---|---|
| *channel handle* | Publisher channel handle pointer |
| *message* | Broadcast message pointer |
| *location* | A number indicating the function in KSR where the process is waiting |

## buffer busy waits

Wait until a buffer becomes available.

There are four reasons that a session cannot pin a buffer in the buffer cache, and a separate wait event exists for each reason:

1. "buffer busy waits": A session cannot pin the buffer in the buffer cache because another session has the buffer pinned.
2. "read by other session": A session cannot pin the buffer in the buffer cache because another session is reading the buffer from disk.
3. "gc buffer busy acquire": A session cannot pin the buffer in the buffer cache because another session is reading the buffer from the cache of another instance.
4. "gc buffer busy release": A session cannot pin the buffer in the buffer cache because another session on another instance is taking the buffer from this cache into its own cache so it can pin it.

Prior to release 10.1, all four reasons were covered by "buffer busy waits." In release 10.1, the "gc buffer busy" wait event covered both the "gc buffer busy acquire" and "gc buffer busy release" wait events.

**Wait Time:** Normal wait time is 1 second. If the session was waiting for a buffer during the last wait, then the next wait will be 3 seconds.

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *id* | The buffer busy waits event is called from different places in the session |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about the potential causes of the buffer busy waits wait event

## buffer deadlock

Oracle does not really wait on this event; the foreground only yields the CPU. Thus, the chances of catching this event are very low. This is not an application induced deadlock, but an assumed deadlock by the cache layer. The cache layer cannot get a buffer in a certain mode within a certain amount of time.

**Wait Time:** 0 seconds. The foreground process only yields the CPU and will usually be placed at the end of the CPU run queue.

| Parameter | Description |
|---|---|
| *class* | See "class" |
| *mode* | See "mode" |
| *flag* | The flag points to the internal flags used by the session to get this block |
| *dba* | See "dba" |

## buffer latch

The session waits on the buffer hash chain latch. Primarily used in the dump routines.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *latch addr* | The virtual address in the SGA where this latch is located. Use the following statement to find the name of this latch:<br>`select *`<br>`from v$latch a, v$latchname b`<br>`where addr = latch addr`<br>`and a.latch# = b.latch#;` |
| *chain#* | The index into array of buffer hash chains. When the chain is 0xffffffff, the foreground waits on the LRU latch. |

## buffer read retry

This event occurs only if the instance is mounted in shared mode (Oracle Real Application Cluster). During the read of the buffer, the contents changed. This means that either:

- The version number, dba, or the incarnation and sequence number stored in the block no longer match
- The checksum on the block does not match the checksum in the block

The block will be reread (this may fail up to 3 times), then corruption is assumed and the corrupt block is dumped in the trace file.

**Wait Time:** The wait time is the elapsed time of the read

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |

## checkpoint completed

A session waits for a checkpoint to complete. This could happen, for example, during a close database or a local checkpoint.

**Wait Time:** 5 seconds

**Parameters:** None

## cleanup of aborted processes

When a process spawn is aborted while the process spawning background is in the middle of spawning, the current session must wait until the pid of the new process is filled in. Once the pid is filled in, then the process spawn can be actually aborted.

**Wait Time:** Usually 3 seconds

| Parameter | Description |
|---|---|
| *location* | Location of the wait |

## control file parallel write

This event occurs while the session is writing physical blocks to all control files. This happens when:

- The session starts a control file transaction (to ensure the control files are up to date in case the session crashes before committing the control file transaction)
- The session commits a transaction to a control file
- Changing a generic entry in the control file, the new value is being written to all control files

**Wait Time:** The wait time is the time it takes to finish all writes to all control files

| Parameter | Description |
|---|---|
| *files* | The number of control files to which the session is writing |
| *blocks* | The number of blocks that the session is writing to the control file |
| *requests* | The number of I/O requests which the session wants to write |

## control file sequential read

Reading from the control file. This happens often. For example, while:

- Making a backup of the control files
- Sharing information (between instances) from the control file
- Reading other blocks from the control files
- Reading the header block

**Wait Time:** The wait time is the elapsed time of the read

| Parameter | Description |
|---|---|
| *file#* | The control file from which the session is reading |
| *block#* | Block number in the control file from where the session starts to read. The block size is the physical block size of the port (usually 512 bytes, some UNIX ports have 1 or 2 Kilobytes). |
| *blocks* | The number of blocks that the session is trying to read |

## control file single write

This wait is signaled while the control file's shared information is written to disk. This is an atomic operation protected by an enqueue (CF), so that only one session at a time can write to the entire database.

**Wait Time:** The wait time is the elapsed time of the write

| Parameter | Description |
|---|---|
| *file#* | This identifies the control file to which the session is currently writing |
| *block#* | Block number in the control file where the write begins. The block size is the as the physical block size of the port (usually 512 bytes, some UNIX ports have 1 or 2 Kilobytes). |
| *blocks* | The number of blocks that the session is trying to write |

## cursor: mutex S

A session waits on this event when it is requesting a mutex in shared mode, when another session is currently holding a this mutex in exclusive mode on the same cursor object.

| Parameter | Description |
|---|---|
| *P1* | Hash value of cursor |

| Parameter | Description |
|---|---|
| *P2* | Mutex value (top 2 bytes contain SID holding mutex in exclusive mode, and bottom two bytes usually hold the value 0) |
| *P3* | Mutex where (an internal code locator) OR'd with Mutex Sleeps |

## cursor: mutex X

The session requests the mutex for a cursor object in exclusive mode, and it must wait because the resource is busy. The mutex is busy because either the mutex is being held in exclusive mode by another session or the mutex is being held shared by one or more sessions. The existing mutex holder(s) must release the mutex before the mutex can be granted exclusively.

| Parameter | Description |
|---|---|
| *P1* | Hash value of cursor |
| *P2* | Mutex value (top 2 bytes contain SID holding mutex in exclusive mode, and bottom two bytes usually hold the value 0) |
| *P3* | Mutex where (an internal code locator) OR'd with Mutex Sleeps |

## cursor: pin S

A session waits on this event when it wants to update a shared mutex pin and another session is currently in the process of updating a shared mutex pin for the same cursor object. This wait event should rarely be seen because a shared mutex pin update is very fast.

**Wait Time:** Microseconds

| Parameter | Description |
|---|---|
| *P1* | Hash value of cursor |
| *P2* | Mutex value (top 2 bytes contains SID holding mutex in exclusive mode, and bottom two bytes usually hold the value 0) |
| *P3* | Mutex where (an internal code locator) OR'd with Mutex Sleeps |

## cursor: pin S wait on X

A session waits for this event when it is requesting a shared mutex pin and another session is holding an exclusive mutex pin on the same cursor object.

**Wait Time:** Microseconds

| Parameter | Description |
|---|---|
| *P1* | Hash value of cursor |
| *P2* | Mutex value (top 2 bytes contains SID holding mutex in exclusive mode, and bottom two bytes usually hold the value 0) |
| *P3* | Mutex where (an internal code locator) OR'd with Mutex Sleeps |

## cursor: pin X

A session waits on this event when it is requesting an exclusive mutex pin for a cursor object and it must wait because the resource is busy. The mutex pin for a cursor object can be busy either because a session is already holding it exclusive, or there are one or more sessions which are holding shared mutex pin(s). The exclusive waiter must wait until all holders of the pin for that cursor object have released it, before it can be granted.

**Wait Time:** Microseconds

| Parameter | Description |
|---|---|
| *P1* | Hash value of cursor |
| *P2* | Mutex value (top 2 bytes contains SID holding mutex in exclusive mode, and bottom two bytes usually hold the value 0) |
| *P3* | Mutex where (an internal code locator) OR'd with Mutex Sleeps |

## Data Guard: process clean up

During Data Guard process termination, Data Guard will wait for one second for process cleanup to complete.

## Data Guard: process exit

During Data Guard process termination, Data Guard will wait for a process to exit before attempting any process cleanup that may be required. It will sleep for one second between each check for process exit.

## Data Guard Broker: single instance

The Data Guard Broker (DMON) process waits for the other instances in this cluster to complete shutdown before continuing with the broker operation.

**Wait Time:** Depends on the number of instances, but not exceeding 30 seconds times the number of instances

## db file asynch I/O submit

When asynchronous I/O is available, this wait event captures any time spent in submitting I/Os to the underlying storage.

**See Also:**

"db file parallel write"

## db file parallel read

This happens during recovery. It can also happen during buffer prefetching, as an optimization (rather than performing multiple single-block reads). Database blocks that must be changed as part of recovery are read in parallel from the database.

**Wait Time:** Wait until all of the I/Os are completed

| Parameter | Description |
|---|---|
| *files* | This indicates the number of files to which the session is reading |
| *blocks* | This indicates the total number of blocks to be read |
| *requests* | This indicates the total number of I/O requests, which will be the same as blocks |

## db file parallel write

This event occurs in the DBWR. It indicates the time that DBWR spends waiting for I/O completion.

If asynchronous I/O is available, then the db file asynch I/O submit wait event captures any time spent in submitting I/Os to the underlying storage.

When asynchronous I/O is not available, db file parallel write captures the time spent during submit and reap.

**Wait Time:** While there are outstanding I/Os, DBWR waits for some of the writes to complete. DBWR does not wait for all of the outstanding I/Os to complete.

| Parameter | Description |
|---|---|
| *requests* | This indicates the total number of I/O requests, which will be the same as blocks |
| *interrupt* | |
| *timeout* | This indicates the timeout value in hundredths of a second to wait for the I/O completion. |

**See Also:**

- *Oracle Database Performance Tuning Guide* about how this wait event can help identify I/O problems
- "db file asynch I/O submit"

## db file scattered read

Similar to **db file sequential read**, except that the session is reading multiple data blocks.

**Wait Time:** The wait time is the actual time it takes to do all of the I/Os

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *blocks* | The number of blocks that the session is trying to read from the *file#* starting at *block#* |

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information about this wait event
- *Oracle Database Performance Tuning Guide* about how this wait event can help identify I/O problems
- *Oracle Database Performance Tuning Guide* about potential causes for this wait event

## db file sequential read

The session waits while a sequential read from the database is performed. This event is also used for rebuilding the control file, dumping data file headers, and getting the database file headers.

**Wait Time:** The wait time is the actual time it takes to do the I/O

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *blocks* | This is the number of blocks that the session is trying to read (should be 1) |

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information about this wait event
- *Oracle Database Performance Tuning Guide* for more information about how this wait event can help identify I/O problems
- *Oracle Database Performance Tuning Guide* about potential causes for this wait event

## db file single write

This event is used to wait for the writing of the file headers.

**Wait Time:** The wait time is the actual time it takes to do the I/O

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *blocks* | This is the number of blocks that the session is trying to write in *file# starting at block#* |

**See Also:**

- *Oracle Database Performance Tuning Guide* about how this wait event can help identify I/O problems

## DFS db file lock

This event occurs only for the DBWR in Real Application Clusters. Each DBWR of every instance holds a global lock on each file in shared mode. The instance that is trying to offline the file will escalate the global lock from shared to exclusive. This signals the other instances to synchronize their SGAs with the control file before the file can be taken offline. The name of this lock is **DF** (see Oracle Enqueue Names for more information).

**Wait Time:** 1 second in loop. The DBWR is waiting in a loop (sleep, check) for the other instances to downgrade to NULL mode. During this time, the DBWR cannot perform other tasks such as writing buffers.

| Parameter | Description |
|---|---|
| *file* | See "file#" |

## DFS lock handle

The session waits for the lock handle of a global lock request. The lock handle identifies a global lock. With this lock handle, other operations can be performed on this global lock (to identify the global lock in future operations such as conversions or release). The global lock is maintained by the DLM.

**Wait Time:** The session waits in a loop until it has obtained the lock handle from the DLM. Inside the loop there is a wait of 0.5 seconds.

| Parameter | Description |
|---|---|
| *name* | See "name and type" |
| *mode* | See "mode" |
| *id1* | See "id1" |
| *id2* | See "id2" |

The session needs to get the lock handle.

## direct path read

During Direct Path operations the data is asynchronously read from the database files. At some stage the session needs to make sure that all outstanding asynchronous I/O have been completed to disk. This can also happen if during a direct read no more slots are available to store outstanding load requests (a load request could consist of multiple I/Os).

**Wait Time:** 10 seconds. The session will be posted by the completing asynchronous I/O. It will never wait the entire 10 seconds. The session waits in a tight loop until all outstanding I/Os have completed.

| Parameter | Description |
|---|---|
| *descriptor address* | This is a pointer to the I/O context of outstanding direct I/Os on which the session is currently waiting |
| *first dba* | The dba of the oldest I/O in the context referenced by the descriptor address |
| *block cnt* | Number of valid buffers in the context referenced by the descriptor address |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## direct path sync

During Direct Path write operations the data is asynchronously written to the database files. At some point the session needs to ensure that all outstanding asynchronous I/O have been completed to disk. On UNIX the `fsync` command, which synchronizes data to disk, is issued to confirm that all the writes have completed and the data is all on disk.

**Wait Time:** The time taken for the fsync operation to complete, which normally is the time taken to complete the outstanding I/Os.

| Parameters | Description |
|---|---|
| *file#* | See "file#" |
| *flags* | Flags used for debugging purposes |

## direct path write

During Direct Path operations, the data is asynchronously written to the database files. At some stage the session needs to make sure that all outstanding asynchronous I/O have been completed to disk. This can also happen if, during a direct write, no more slots are available to store outstanding load requests (a load request could consist of multiple I/Os).

**Wait Time:** 10 seconds. The session will be posted by the completing asynchronous I/O. It will never wait the entire 10 seconds. The session waits in a tight loop until all outstanding I/Os have completed.

| Parameter | Description |
|---|---|
| *descriptor address* | This is a pointer to the I/O context of outstanding direct I/Os on which the session is currently waiting |
| *first dba* | The dba of the oldest I/O in the context referenced by the descriptor address |
| *block cnt* | Number of valid buffers in the context referenced by the descriptor address |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this parameter

## Disk file operations I/O

This event is used to wait for disk file operations (for example, open, close, seek, and resize). It is also used for miscellaneous I/O operations such as block dumps and password file accesses.

**Wait Time:** The wait time is the actual time it takes to do the I/O

| Parameter | Description |
|---|---|
| *FileOperation* | Type of file operation |
| *fileno* | File identification number |
| *filetype* | Type of file (for example, log file, data file, and so on) |

## dispatcher shutdown

During shutdown immediate or normal, the shutdown process must wait for all the dispatchers to shutdown. As each dispatcher is signaled, the session that causes the shutdown is waits on this event until the requested dispatcher is no longer alive.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *waited* | Indicates the cumulative wait time. After 5 minutes, the session writes to the alert and trace files to indicate that there might be a problem. |

**See Also:**

*Oracle Database SQL Language Reference* for information about shutting down a dispatcher using `SHUTDOWN` clause of the SQL `ALTER SYSTEM` statement

## dispatcher timer

This basically means that the dispatcher is idle and waiting for some work to arrive.

**Wait Time:** 60 seconds

| Parameter | Description |
|---|---|
| *sleep time* | The intended sleep time. The dispatcher will return to work sooner if it is posted by either data arriving on the network or by a post from a shared server process to send data back to the client. |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## duplicate cluster key

It is possible for a race condition to occur when creating a new cluster key. If it is found that another process has put the cluster key into the data/index block, then the session waits and retries. The retry should then find a valid cluster key.

**Wait Time:** 0.01 seconds

| Parameter | Description |
|---|---|
| *dba* | The dba of the block into which the session is trying to insert a cluster key |

## enq: OW - initialization

A session will wait on this event if it is trying to initialize the database wallet, and another session has already begun an initialization.

**Wait Time:** Total time necessary to initialize the wallet context

**Parameters:** None

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about the potential causes of an enqueue wait event

## enq: OW - termination

A session will wait on this event if it is trying to terminate the database wallet, and another session has already begun a termination.

**Wait Time:** Total time necessary to deallocate memory used by the wallet context and terminate the context.

**Parameters:** None

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about the potential causes of an enqueue wait event

## enqueue

The session is waiting for a local enqueue. The wait depends on the name of the enqueue (see Oracle Enqueue Names).

**Wait Time:** Depends on the enqueue name

| Parameter | Description |
|---|---|
| *name* | See "name and type" |
| *mode* | See "mode" |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about the potential causes of an enqueue wait event

## flashback buf free by RVWR

This wait event only occurs when Flashback Database is turned on. A session waits for recovery writer (RVWR) to write flashback data to the flashback logs on disk because the buffers are full. Until RVWR can free up the buffers, the session may need to wait.

If this event becomes a top wait event for the database, it is typically because the file system or storage system for the Fast Recovery Area does not support enough bandwidth for Oracle to write the flashback database logs. Refer to the Flashback Database section in *Oracle Database Backup and Recovery User's Guide* olink:BRADVfor tuning considerations.

**Wait Time:** 1 second

**Parameters:** None

## flashback logfile sync

Waits for flashback database data to be written to disk.

**Wait Time:** Includes RVWR writing the flashback database data and posting this process

**Parameters:** None

## free buffer waits

This will happen if:

- All buffer gets have been suspended. This could happen when a file was read-only and is now read/write. All the existing buffers must be invalidated since they are not linked to lock elements (needed when mounted parallel (shared)). So cache buffers are not assigned to data block addresses until the invalidation is finished.
- The session moved some dirty buffers to the dirty queue and now this dirty queue is full. The dirty queue must written first. The session will wait on this event and try again to find a free buffer
- This also happens after inspecting **free buffer inspected** buffers. If no free buffer is found, Oracle waits for one second, and then tries to get the buffer again (depends on the context). For more information, see free buffer inspected.

**Wait Time**: 1 second

| Parameter | Description |
|---|---|
| *file#* | See "file#" |

| Parameter | Description |
|---|---|
| *block#* | See "block#" |

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information about this wait event
- *Oracle Database Performance Tuning Guide* about potential causes of this wait event

## free global transaction table entry

The session is waiting for a free slot in the global transaction table (used by the Distributed Database option). It will wait for 1 second and try again.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *tries* | The number of times the session tried to find a free slot in the global transaction table |

## free process state object

Used during the creation of a process. The session will scan the process table and look for a free process slot. If none can be found, PMON is posted to check if all the processes currently in the process table are still alive. If there are dead processes, then PMON will clean them and make the process slot available to new processes. The waiting process will then rescan the process table to find the new slot.

**Wait Time:** 1 second

**Parameters:** None

## gc recovery quiesce

Instance recovery is waiting for a global cache operation to complete.

**Wait Time:** The total elapsed time for the global cache operation to complete

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *class* | See "class" |

## GCS lock open S

The session waits for a resource get in SHARED mode on the block identified by file# and block#.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *class* | See "class" |

## GCS lock open X

The session waits for a resource get in `EXCLUSIVE` mode on the block identified by file# and block#.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *lenum* | See "le" |

## gcs remastering wait for drop pkey

A session dropping an object waits on the lock manager daemon (LMD) to remove the object's affinity to an instance.

Wait Time: 20 hundredths of a second

| Parameter | Description |
|---|---|
| *pkey* | The object id of the database object being dropped |

## global cache busy

The session waits to convert a buffer from Shared Current to Exclusive Current status.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *le* | See "lenum" on page A-8 |

## global cache lock cleanup

PMON is waiting for an LCK process to cleanup the lock context after a foreground process died while doing a global cache lock operation.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |

| Parameter | Description |
|---|---|
| *lenum* | See "lenum" on page A-8 |

## global cache freelist

All releasable locks are used and a new one has been requested. To make a resource element available, a resource element is pinged.

**Wait Time:** The duration of the resource get operation to ping the resource element

**Parameters:** None

## inactive session

This event is used for two purposes:

- Switching sessions

  If a timeout period has been specified, then wait that amount of time for the session to be detached.

- Killing sessions

  From either `KILL SESSION` or internal request. Having posted a session that it should kill itself, wait for up to 1 minute for the session to terminate.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *session#* | See "session#" |
| *waited* | See "waited" |

## inactive transaction branch

The session waits for a transaction branch that is currently used by another session.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *branch#* | The serial number of the transaction for which the session is waiting |
| *waited* | See "waited" |

## index block split

While trying to find an index key in an index block, Oracle noticed that the index block was being split. Oracle will wait for the split to finish and try to find the key again.

**Wait Time:** The session will yield the CPU, so there is no actual waiting time

| Parameter | Description |
|---|---|
| *rootdba* | The root of the index |

| Parameter | Description |
|---|---|
| *level* | This is the level of the block that the session is trying to split in the index. The leaf blocks are level 0. If the level is > 0, it is a branch block. (The root block can be considered a special branch block). |
| *childdba* | The block that the session is trying to split |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about index block splits

## instance state change

The session waits for SMON to enable or disable cache or transaction recovery. This usually happens during `ALTER DATABASE OPEN` or `CLOSE`.

**Wait Time:** Wait time depends on the amount of time the action takes (that is, the amount of recovery needed)

| Parameter | Description |
|---|---|
| *layer* | This value can be 1 or 2. If 1, it means that the transaction layer wants transaction recovery to be performed. If 2, it means that cache recovery will be performed. |
| *value* | This value can be 0 (disable) or 1 (enable) |
| *waited* | The number of seconds waited so far |

## io done

The session waits for an I/O to complete or it waits for a slave process to become available to submit the I/O request. This event occurs on platforms that do not support asynchronous I/O.

**Wait Time:** 50 milliseconds

| Parameter | Description |
|---|---|
| *msg ptr* | A pointer to the I/O request |

## kcl bg acks

The session waits for the background LCK process(es) to finish what they are doing. For example:

- Lock recovery
- Initializing the locks (start up)
- Finalizing the locks (shut down)

**Wait Time:** 10 seconds

| Parameter | Description |
|---|---|
| *count* | The number of LCK processes that have finished |
| *loops* | The number times the process had to wait for the LCK processes to finish what they were doing |

## ksxr wait for mount shared

The cross instance broadcast facility of this Oracle instance is waiting for the database mount in shared mode to complete.

**Wait Time:** The time taken for the instance to mount. An indefinite wait on this event implies that the instance startup is hanging.

## ktm: instance recovery

The session waits for SMON to finish the instance, transaction recovery, or sort segment cleanup.

**Wait Time:** The wait time can vary and depends on the amount of recovery needed

| Parameter | Description |
|---|---|
| *undo segment#* | If the value is 0, SMON is probably performing instance recovery. If P1 > 0, use this query to find the undo segment:<br>`select *`<br>`from v$rollstat`<br>`where usn = undo segment#;` |

## latch activity

This event is used as part of the process of determining whether a latch must be cleaned.

**Wait Time:** 0.05 to 0.1 seconds

| Parameter | Description |
|---|---|
| *address* | The address of the latch that is being checked |
| *number* | The latch number of the latch that has activity. To find more information on the latch, use the following SQL statement:<br>`select *`<br>`from v$latchname`<br>`where latch# = number;` |
| *process#* | If this is 0, it is the first phase of the in-flux tests |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about latch wait events

## latch free

The process waits for a latch that is currently busy (held by another process).

**Wait Time:** The wait time increases exponentially and does not include spinning on the latch (active waiting). The maximum wait time also depends on the number of latches that the process is holding. There is an incremental wait of up to 2 seconds.

| Parameter | Description |
|---|---|
| *address* | The address of the latch for which the process is waiting |
| *number* | The latch number that indexes in the V$LATCHNAME view.To find more information on the latch, use the following SQL statement:<br>`select *`<br>`from v$latchname`<br>`where latch# = number;` |
| *tries* | A count of the number of times the process tried to get the latch (slow with spinning) and the process has to sleep |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about latch wait events

## library cache load lock

The session tries to find the load lock for the database object so that it can load the object. The load lock is always obtained in Exclusive mode, so that no other process can load the same object. If the load lock is busy the session will wait on this event until the lock becomes available.

**Wait Time:** 3 seconds (1 second for PMON)

| Parameter | Description |
|---|---|
| *object address* | Address of the object being loaded |
| *lock address* | Address of load lock being used |
| *mask* | Indicates which data pieces of the object that must loaded |

## library cache lock

This event controls the concurrency between clients of the library cache. It acquires a lock on the object handle so that either:

- One client can prevent other clients from accessing the same object
- The client can maintain a dependency for a long time (for example, no other client can change the object)

This lock is also obtained to locate an object in the library cache.

**Wait Time:** 3 seconds (1 second for PMON)

| Parameter | Description |
|---|---|
| *handle address* | Address of the object being loaded |
| *lock address* | Address of the load lock being used. This is not the same thing as a latch or an enqueue, it is a State Object. |
| *mode* | Indicates the data pieces of the object which must be loaded |
| *namespace* | See "namespace" |

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information about this wait event
- *Oracle Database Performance Tuning Guide* for more information about potential causes of this wait event

## library cache pin

This event manages library cache concurrency. Pinning an object causes the heaps to be loaded into memory. If a client wants to modify or examine the object, the client must acquire a pin after the lock.

**Wait Time:** 3 seconds (1 second for PMON)

| Parameter | Description |
|---|---|
| *handle address* | Address of the object being loaded |
| *pin address* | Address of the load lock being used. This is not the same thing as a latch or an enqueue, it is basically a State Object. |
| *mode* | Indicates which data pieces of the object that must be loaded |
| *namespace* | See "namespace" |

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information about this wait event
- *Oracle Database Performance Tuning Guide* for more information about potential causes of this wait event

## library cache shutdown

The process shutting down the instance waits for sessions to complete before proceeding with library cache shutdown.

## LMON global data update

The rolling migration operation is waiting for a response from LMON to acknowledge the global data was updated.

**Wait Time:** The time it takes for LMON to publish/retrieve the global data associated with a

**Parameters:** None

## lock manager wait for remote message

The lock manager waits for a message from a remote lock manager in the same configuration.

**Wait Time:** The elapsed time of the wait

| Parameter | Description |
|---|---|
| *waittime* | The elapsed time of the actual wait |

## Log archive I/O

Used local archiving of online redo logs (for a production database) or standby redo logs (for a standby database). When the archiving process exhausts its I/O buffers because all of them are being used for on-going I/O's, the wait for an available I/O buffer is captured in this system wait event.

**Wait Time:** Depends on the speed of the disks

**Parameters:** None

## log buffer space

Waiting for space in the log buffer because the session is writing data into the log buffer faster than LGWR can write it out. Consider making the log buffer bigger if it is small, or moving the log files to faster disks such as striped disks.

**Wait Time:** Usually 1 second, but 5 seconds if it is waiting for a Switch Logfile to complete

**Parameters:** None

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information about this wait event
- *Oracle Database Performance Tuning Guide* for more information about potential causes of this wait event

## log file parallel write

Writing redo records to the redo log files from the log buffer.

**Wait Time:** Time it takes for the I/Os to complete. Even though redo records are written in parallel, the parallel write is not complete until the last I/O is on disk.

| Parameter | Description |
|---|---|
| *files* | Number of files to be written |
| *blocks* | Number of blocks to be written |

| Parameter | Description |
|---|---|
| *requests* | Number of I/O requests |

**See Also:**

*Oracle Database Performance Tuning Guide* for information about how this wait event can help identify I/O problems

## log file sequential read

Waiting for the read from this logfile to return. This is used to read redo records from the log file.

**Wait Time:** Time it takes to complete the physical I/O (read)

| Parameter | Description |
|---|---|
| *log#* | The relative sequence number of the logfiles within a log group (used only when dumping the logfiles) |
| *block#* | See "block#" |
| *blocks* | The number of blocks to read |

## log file single write

Waiting for the write to this logfile to complete. This event is used while updating the header of the logfile. It is signaled when adding a log file member and when incrementing sequence numbers.

**Wait Time:** Time it takes for the physical I/O (write) to complete

| Parameter | Description |
|---|---|
| *log#* | This is the number of the group/log to which the session is currently writing |
| *block#* | See "block#" |
| *blocks* | The number of blocks to write |

## log file switch (archiving needed)

Waiting for a log switch because the log that the LGWR will be switching into has not been archived yet. Check the alert log to ensure that archiving has not stopped due to a failed archive write. To speed archiving, consider adding more archive processes or putting the archive files on striped disks.

**Wait Time:** 1 second

**Parameters:** None

## log file switch (checkpoint incomplete)

Waiting for a log switch because the session cannot wrap into the next log. Wrapping cannot be performed because the checkpoint for that log has not completed.

**Wait Time:** 1 second

**Parameters:** None

## log file switch (clearing log file)

Waiting for a log switch because the log is being cleared due to a `CLEAR LOGFILE` command or implicit clear logfile executed by recovery.

**Wait Time:** 1 second

**Parameters:** None

## log file switch (private strand flush incomplete)

User sessions trying to generate redo, wait on this event when LGWR waits for DBWR to complete flushing redo from IMU buffers into the log buffer; when DBWR is complete LGWR can then finish writing the current log, and then switch log files.

**Wait Time:** 1 second

**Parameters:** None

## log file switch completion

Waiting for a log switch to complete.

**Wait Time:** 1 second

**Parameters:** None

## log file sync

When a user session commits, the session's redo information must be flushed to the redo logfile. The user session will post the LGWR to write the log buffer to the redo log file. When the LGWR has finished writing, it will post the user session.

**Wait Time:** The wait time includes the writing of the log buffer and the post.

| Parameter | Description |
|---|---|
| *buffer#* | The number of the physical buffer in the redo log buffer that must be synchronized |

**See Also:**

- *Oracle Database Performance Tuning Guide* for more information about this wait event
- *Oracle Database Performance Tuning Guide* for information about potential causes of this wait event

## log switch/archive

Used as part of the `ALTER SYSTEM ARCHIVE LOG CHANGE` *scn* statement. The session waits for the current log from all open threads to be archived.

**Wait Time:** Wait for up to 10 seconds

| Parameter | Description |
|---|---|
| *thread#* | The thread number of the thread that is currently archiving its current log |

## optimizer stats update retry

When concurrent sessions try to update optimizer statistics for the same object, all of them except the one that successfully acquired all necessary locks/pins on the library/row cache entries, wait on this wait event and then retry locking after a short period of time. In addition to explicit statistics gathering and maintenance operations using the DBMS_STATS package, Oracle database itself may try to update statistics for some objects, either on behalf of the user or for its own maintenance purposes.

**Wait Time:** 10 ms

**Parameters:** None

## parallel recovery change buffer free

The parallel recovery coordinator is waiting for a change mapping buffer to be released by one of the recovery slaves.

**Wait Time:** 100ms

**Parameters:** None

## parallel recovery control message reply

The parallel recovery coordinator is waiting for all recovery slaves to respond to a synchronous control message.

**Wait Time:** 100ms

**Parameters:** None

## parallel recovery coord send blocked

The parallel recovery coordinator cannot send a redo change message because the recovery slave is still actively applying redo that it has already received and has not yet released the channel.

**Wait Time:** 100ms

**Parameters:** None

## parallel recovery coord wait for reply

The parallel recovery coordinator is waiting for all recovery slaves to exit.

**Wait Time:** 100ms

**Parameters:** None

## parallel recovery coordinator waits for slave cleanup

The parallel recovery coordinator is waiting for all recovery slaves to exit gracefully.

**Wait Time:** 10ms

**Parameters:** None

## parallel recovery read buffer free

The parallel recovery coordinator is waiting for a log read buffer to be released by all recovery slaves. Only after every recovery slave finishes applying redo from a log read buffer, can the buffer be used by the coordinator to issue the next log read.

**Wait Time:** 100ms

**Parameters:** None

## parallel recovery slave next change

A parallel recovery slave is idle and waiting for the next change message from the coordinator.

**Wait Time:** 100ms

**Parameters:** None

## pending global transaction(s)

This event should happen only during testing. The session waits for pending transactions to clear.

**Wait Time:** 30 seconds

| Parameter | Description |
|---|---|
| *scans* | Number of times the session has scanned the `PENDING_TRANS$` table |

## pipe get

The session waits for a message to be received on the pipe or for the pipe timer to expire.

**Wait Time:** There is a 5 second wake up (check) and the pipe timer set by the user

| Parameter | Description |
|---|---|
| *handle address* | The library cache object handle for this pipe |
| *buffer length* | The length of the buffer |
| *timeout* | The pipe timer set by the user |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## pipe put

The session waits for the pipe send timer to expire or for space to be made available in the pipe.

**Wait Time:** There is the 5 second wakeup (check) and the user-supplied timeout value

| Parameter | Description |
|---|---|
| *handle address* | The library cache object handle for this pipe |
| *record length* | The length of the record or buffer that has been put into the pipe |
| *timeout* | The pipe timer set by the user |

## PL/SQL lock timer

This event is called through the `DBMS_LOCK.SLEEP` procedure. This event will most likely originate from procedures written by a user.

**Wait Time:** The wait time is in hundredths of seconds and depends on the user context

| Parameter | Description |
|---|---|
| *duration* | The duration that the user specified in the `DBMS_LOCK.SLEEP` procedure |

## pmon timer

This is the main wait event for PMON. When PMON is idle, it is waiting on this event.

**Wait Time:** Up to 3 seconds, if not posted before

| Parameter | Description |
|---|---|
| *duration* | The actual amount of time that the PMON is trying to sleep |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## prewarm transfer retry

Release a hash latch, then wait under this event before attempting to re-acquire the hash latch.

**Wait Time:** 10ms

**Parameters:** None

## prior process spawner to be cleaned up

When a prior process has died while spawning a background, the current process which is trying to spawn new a background must wait until the prior process state is cleaned up.

**Wait Time:** Usually 3 - 10 seconds

| Parameter | Description |
|---|---|
| *process_pid* | process identifier (see `V$PROCESS.PID`) of the process whose state must be cleaned up. |

| Parameter | Description |
|---|---|
| *process_sno* | process serial number (see V$PROCESS.SERIAL#) of the process whose state must be cleaned up. |

## process startup

Wait for a shared server, Dispatcher, or other background process to start.

**Wait Time:** Wait up to 1 second for a background process to start. If timed out, then re-wait until 5 minutes have passed and signal an error. If the process has started, the event will acknowledge this.

| Parameter | Description |
|---|---|
| *type* | The process type that was started |
| *process#* | The process number of the process being started |
| *waited* | Cumulative time waited for the process to start |

## PX Deque wait

The process is waiting for a message during a parallel execute.

**Wait Time:** The wait time depends on how quickly the message arrives. Wait times can vary, but it will normally be a short period of time.

| Parameter | Description |
|---|---|
| *reason* | The reason for dequeuing |
| *sleeptime* | The amount of time that the session slept |
| *loop* | The total number of times that the session has slept |

## PX qref latch

Each parallel execution process has a parallel execution qref latch, which must be acquired before the queue buffers can be manipulated.

**Wait Time:** Wait up to 1 second

| Parameter | Description |
|---|---|
| *function* | Indicates the type of wait that the session is doing |
| *sleeptime* | The amount of time that the session waits (in hundredths of a second) |
| *qref* | The address of the process queue for which the session is waits |

## PX server shutdown

During normal or immediate shutdown the parallel execution slaves are posted to shutdown cleanly. If any parallel execution slaves are still alive after 10 seconds, they are killed.

**Wait Time:** Wait up to 0.5 seconds

| Parameter | Description |
|---|---|
| *nalive* | The number of parallel execution slaves that are still running |
| *sleeptime* | The total sleeptime since the session started to wait on this event |
| *loop* | The number of times the session waited for this event |

## PX signal server

This event occurs only in Exclusive mode. The query coordinator is signalling the Query Slaves that an error has occurred.

**Wait Time:** 0.5 seconds

| Parameter | Description |
|---|---|
| *serial* | The serial number of the slave process queue |
| *error* | The error that has occurred |
| *nbusy* | The number of slave processes that are still busy |

## rdbms ipc message

The background processes (LGWR, DBWR, LMS0) use this event to indicate that they are idle and are waiting for the foreground processes to send them an IPC message to do some work.

**Wait Time:** Up to 3 seconds. The parameter *timeout* shows the true sleep time.

| Parameter | Description |
|---|---|
| *timeout* | The amount of time that the session waits for an IPC message |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## rdbms ipc message block

This event indicates that all message blocks are in use and that the session had to wait for a message block to become available.

**Wait Time:** Wait up to 60 seconds

**Parameters:** None

## rdbms ipc reply

This event is used to wait for a reply from one of the background processes.

**Wait Time:** The wait time is specified by the user and is indicated by the parameter *timeout*.

| Parameter | Description |
|---|---|
| *from_process* | The background process for which the session is waiting. The wait is for a reply to an IPC message sent by the session. |
| *timeout* | The amount of time in seconds that this process will wait for a reply |

## read by other session

This event occurs when a session requests a buffer that is currently being read into the buffer cache by another session. Prior to release 10.1, waits for this event were grouped with the other reasons for waiting for buffers under the 'buffer busy waits' event

**Wait Time:** Time waited for the buffer to be read by the other session (in microseconds)

| Parameter | Description |
|---|---|
| *file#* | See "file#" |
| *block#* | See "block#" |
| *class#* | See "class" |

## recovery read

A parallel recovery slave (or serial recovery process) is waiting for a batch of data block reads to complete.

**Wait Time:** Time it takes to complete the physical I/O (read)

**Parameters:** None

## Redo Transport Attach

Used to track the time spent (in centi-seconds) doing Connect, Logon, and RFSATTACH for *any* network process.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about using RFS to manage standby redo logs.

## Redo Transport Close

Used to track the time spent (in centi-seconds) by ARC*n*, NSS*n*, and TT*nn* processes doing RFSCLOSE and RFSRGSTR operations.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about using RFS to manage standby redo logs.

## Redo Transport Detach

Used to track the time spent (in centi-seconds) doing RFSDETACH and Disconnect for *any* network process.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about using RFS to manage standby redo logs.

## Redo Transport Open

Used to track the time spent (in centi-seconds) by ARC*n*, NSS*n*, and TT*nn* background processes doing RFSCREAT and RFSANNCE operations.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about using RFS to manage standby redo logs.

## Redo Transport Ping

Used to track the time spent (in centi-seconds) by ARC*n* background processes doing RFSPING operations.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about using RFS to manage standby redo logs.

## Redo Transport Slave Shutdown

Used to track the time spent (in centi-seconds) by LGWR doing NSS*n* and TT*nn* process shutdown and termination.

## Redo Transport Slave Startup

Used to track the time spent (in centi-seconds) by LGWR doing NSS*n* and TT*nn* process startup and initialization.

## Redo Writer Remote Sync Complete

Used to track the time spent (in centi-seconds) by LGWR reaping completed network writes to remote destinations.

## Redo Write Remote Sync Notify

Used to track the time spent (in centi-seconds) by LGWR issuing network writes to remote destinations.

## Remote Sync Ping

Used to track the time spent (in centi-seconds) by the LGWR and NSS*n* background processes doing synchronous PING operations.

## resmgr:become active

The session is waiting for a resource manager active session slot. This event occurs when the resource manager is enabled and the number of active sessions in the session's current consumer group exceeds the current resource plan's active session limit for the consumer group. To reduce the occurrence of this wait event, increase the active session limit for the session's current consumer group.

**Wait Time:** The time the session waited to be allocated an active session slot

| Parameter | Description |
|---|---|
| *location* | location of the wait |

## resmgr:cpu quantum

The session is waiting to be allocated a quantum of cpu. This event occurs when the resource manager is enabled and is throttling CPU consumption. To reduce the occurrence of this wait event, increase the CPU allocation for the sessions's current consumer group.

Wait Time: The time the session waited to acquire a CPU quantum

| Parameter | Description |
|---|---|
| *location* | Location of the wait |

## resmgr:pq queued

The session is waiting in the parallel statement queue.

Wait Time: The time the session waited for sufficient parallel query processes to become available to run this session with the requested degree of parallelism

| Parameter | Description |
|---|---|
| *location* | Location of the wait |

**See Also:**

*Oracle Database VLDB and Partitioning Guide* for more information about this wait event

## rolling migration: cluster quiesce

This is the wait event that instances wait on when cluster is about to start a rolling migration. The instances are waiting for any privileged operations that blocks rolling migration to complete before allowing rolling migration.

Wait Time: 1 second

| Parameter | Description |
|---|---|
| *location* | Its value will be 1 if the wait is for completion of the privileged operations so that a rolling upgrade/downgrade can start. Its value will be 2 if the wait is for completion of the rolling upgrade/ downgrade on all the nodes in the cluster. |
| *waits* | The number of seconds spent waiting at the current location. |

## row cache lock

The session is trying to get a data dictionary lock.

**Wait Time:** Wait up to 60 seconds.

| Parameter | Description |
|---|---|
| *cache id* | The CACHE# column value in the V$ROWCACHE view |
| *mode* | See "mode" |
| *request* | The pipe timer set by the user |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## RVWR wait for flashback copy

Waits for a process to copy flashback database data into the flashback buffer, in order to write out the requested flashback data.

**Wait TIme:** 10 milliseconds

| Parameter | Description |
|---|---|
| *copy latch #* | The number of flashback copy latch RVWR is waiting on |

## sbtbufinfo

This function is called when Oracle needs to discover the size, and number, of I/O buffers that have been allocated by the SBT layer. It should be very fast and never block.

**Wait Time:** Less than one millisecond

**Parameters:** None

## sbtgetbuf

This function obtains one I/O buffer that Oracle will use for I/O during a backup job.

**Wait Time:** Less than one millisecond

**Parameters:** None

## sbtmapbuf

This is an internal function used to facilitate multi-process buffer management. It should be very fast and never block.

**Wait Time:** Less than one millisecond

**Parameters:** None

## sbtrelbuf

This function releases an I/O buffer that has been already processed during a restore job, so that the SBT layer can fill it with more data. It should be very fast and never block.

**Wait Time:** Less than one millisecond

**Parameters:** None

## scginq AST call

Called by the session to find the highest lock mode that is held on a resource.

**Wait Time:** Wait up to 0.2 seconds, but the wait will continue until the NULL mode Acquisition AST has fired.

**Parameters:** None

## SGA: allocation forcing component growth

Process waiting on an immediate mode memory transfer with auto-tune SGA after a 4031 for MMAN to get the memory and post it.

**Wait Time:** 10 msec

**Parameters:** None

## SGA: MMAN sleep for component shrink

MMAN to wait and post itself for satisfying an auto-tuned memory request while trying to fully free a component's quiesced granules. In Release 10.1, the name of this event was 'wait for SGA component shrink'.

**Wait Time:** 10 msec

| Parameter | Description |
|---|---|
| *P1* | component_id (corresponding to the memory pool) |
| *P2* | Current size in granules |
| *P3* | Target size in granules |

## SGA: sga_target resize

Memory resize requests wait while sga target is being resized. In Release 10.1, the name of this event was 'wait for sga_target resize'.

**Wait Time:** 10 msec

**Parameters:** None

## Shared IO Pool Memory

Wait until a shared I/O pool buffer becomes available. This happens when processes are using these buffers for I/O and the current process needs to wait for the release of any one of the buffers to the shared I/O pool.

**Wait Time:** 10msec

**Parameters:** None

## shared server idle wait

Idle wait event for a shared server. The server waits on the common queue for a virtual circuit. (See also "virtual circuit wait".)

**Wait Time:** 30 seconds

**Parameters:** None

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## single-task message

When running single task, this event indicates that the session waits for the client side of the executable.

**Wait Time:** Total elapsed time that this session spent in the user application

**Parameters:** None

## smon timer

This is the main idle event for SMON. SMON will be waiting on this event most of the time until it times out or is posted by another process.

**Wait Time:** 5 minutes (300 seconds)

| Parameter | Description |
|---|---|
| *sleeptime* | The amount of time that SMON tries to wait on this event in seconds |
| *failed* | The number of times SMON was posted when there some kind of error |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## SQL*Net break/reset to client

The server sends a break or reset message to the client. The session running on the server waits for a reply from the client.

**Wait Time:** The actual time it takes for the break or reset message to return from the client

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *break?* | See "break?" |

## SQL*Net break/reset to dblink

Same as **SQL*Net break/reset to client**, but in this case, the break/reset message is sent to another server process over a database link.

**Wait Time:** The actual time it takes for the break or reset message to return from the other server process

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *break?* | See "break?" |

## SQL*Net message from client

The server process (foreground process) waits for a message from the client process to arrive.

**Wait Time:** The time it took for a message to arrive from the client since the last message was sent to the client

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes received by the server (foreground process) from the client. |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## SQL*Net message from dblink

The session waits while the server process (foreground process) receives messages over a database link from another server process.

**Wait Time:** The time it took for a message to arrive from another server (foreground process) since a message was sent to the other foreground process.

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes received by the server (foreground process) from another foreground process over a database link. |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## SQL*Net message to client

The server (foreground process) is sending a message to the client.

**Wait Time:** The actual time the **send** takes

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes sent by the server process to the client |

## SQL*Net message to dblink

The server process (foreground process) is sending a message over a database link to another server process.

**Wait Time:** The actual time the **send** takes

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes sent by the server process to another server process over a database link |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## SQL*Net more data from client

The server is waiting on the client to send more data to its client shadow process, in an already initiated operation.

**Wait Time:** The time waited depends on the time it took to receive the data (including the waiting time)

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes received from the client |

## SQL*Net more data from dblink

The foreground process is expecting more data from a data base link.

**Wait Time:** The total time it takes to read the data from the database link (including the waiting time for the data to arrive)

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes received |

## SQL*Net more data to client

The server process is sending more data/messages to the client. The previous operation to the client was also a **send**.

**Wait Time:** The actual time it took for the **send** to complete

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes that are being sent to the client |

**See Also:**

*Oracle Database Performance Tuning Guide* for more information about this wait event

## SQL*Net more data to dblink

The event indicates that the server is sending data over a database link again. The previous operation over this database link was also a **send**.

**Wait Time:** The actual time it takes to send the data to the other server

| Parameter | Description |
|---|---|
| *driver id* | See "driver id" |
| *#bytes* | The number of bytes that are sent over the database link to the other server process |

## Streams AQ: waiting for messages in the queue

The session is waiting on an empty OLTP queue (Advanced Queuing) for a message to arrive so that the session can dequeue that message.

**Wait Time:** The amount of time that the session wants to wait is determined by the parameter *wait time*

| Parameter | Description |
|---|---|
| *queue id* | The ID of the OLTP queue for which this session is waiting |
| *process#* | The process number of the process in which this session runs |
| *wait time* | The intended wait time for this session |

## switch logfile command

The session waits on the user command `SWITCH LOGFILE` to complete.

**Wait Time:** 5 seconds

**Parameters:** None

## SYNC Remote Write

Used to track the time spent by LGWR doing SYNC RFSWRITE operations.

**See Also:**

*Oracle Data Guard Concepts and Administration* for more information about using RFS to manage standby redo logs.

## TCP Socket (KGAS)

A session is waiting for an external host to provide requested data over a network socket. The time that this wait event tracks does not indicate a problem, and even a long wait time is not a reason to contact Oracle Support. It naturally takes time for data to flow between hosts over a network, and for the remote aspect of an application to process any request made to it. An application that communicates with a remote host must wait until the data it will read has arrived. In addition, on Microsoft Windows, a separate thread monitors the arrival of traffic. This thread spends most of its life in waits tracked by the TCP Socket (KGAS) wait event.

**Wait Time:** The total elapsed time for the network connection to be established or for data to arrive from over the network

| Parameter | Description |
|---|---|
| *P0* | For Oracle internal use only. Values 8, 9, and 10 occur within the special thread present on Microsoft Windows; other P0 values occur in normal user sessions. |
| *P1* | For Oracle internal use only |

## timer in sksawat

The session waits for the Archiver (ARC*n*) asynchronous I/O to complete.

**Wait Time:** 0.01 seconds

**Parameters:** None

## transaction

Wait for a blocking transaction to be rolled back. Continue waiting until the transaction has been rolled back.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *undo seg#* | The rollback segment ID |
| *slot#* | The slot ID inside the rollback segment |
| *wrap#* | The sequence number that is incremented for each transaction |

| Parameter | Description |
|---|---|
| *count* | The number of times that the session has waited on this transaction |

## unbound tx

The session waits to see if there are any transactions that have been started but do not have a Rollback Segment associated with them.

**Wait Time:** 1 second

**Parameters:** None

## undo_retention publish retry

This wait can occur for two reasons. A session issuing an `ALTER SYSTEM SET UNDO_RETENTION` may wait on this event wait while a cluster reconfiguration takes place. Or the background process MMNL may wait for cluster reconfiguration while attempting to determine the max `UNDO_RETENTION`.

**Wait time:** 1 second

| Parameter | Description |
|---|---|
| *P1* | Identifies where the retry is happening. Id = 1 retry while publishing into the max undo_retention namespace. Id = 2 retry while iterator accessing the max undo_retention namespace |
| P2 | Retry count (maximum number of retries is 5) |

## undo segment extension

The undo segment is being extended or shrunk. The session must wait until the operation on the undo segment has finished.

**Wait Time:** 0.01 seconds

| Parameter | Description |
|---|---|
| *segment#* | The ID of the rollback segment that is being extended or shrunk |

## undo segment recovery

PMON is rolling back a dead transaction. The wait continues until rollback finishes.

**Wait Time:** 3 seconds

| Parameter | Description |
|---|---|
| *segment#* | The ID of the rollback segment that contains the transaction that is being rolled back |
| *tx flags* | The transaction flags (options) set for the transaction that is being rolled back |

## undo segment tx slot

Wait for a transaction slot to become available within the selected rollback segment. Continue waiting until the slot is available.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *segment#* | The ID of the rollback segment that contains the transaction that is being rolled back |

## utl_file I/O

Waits associated with operations performed using the `UTL_FILE` package.

## virtual circuit status

The session waits for a virtual circuit to return a message type indicated by status.

**Wait Time:** 30 seconds

| Parameter | Description |
|---|---|
| *circuit#* | Indicates the virtual circuit# being waited on |
| *status* | Indicates what the session is waiting for |

**Note:**

The virtual circuit status wait event has been deprecated. It is replaced by the virtual circuit wait event.

## virtual circuit wait

The session waits for a virtual circuit operation to complete.

**Wait Time:** 30 seconds

| Parameter | Description |
|---|---|
| *circuit#* | Indicates the virtual circuit# being waited on |
| *type* | Indicates the type of operation the session is waiting for |

## WCR: replay client notify

During replay, the Workload Replay Client always keeps an open connection to the database to detect some special errors. This session will normally be in the wait state until some specific replay-related exception occurs.

**Wait Time:** Waits until notified. Times out every 30 seconds.

| Parameter | Description |
|---|---|
| *who am I* | Identifies the reason for wait for the admin thread.<br>1 - waiting for REPLAY.START() to be issued.<br>2 - waiting for the replay to run to completion or REPLAY.CANCEL() to be issued. |

## WCR: replay clock

A session will wait on this event during replay if it has some logical dependencies on another session that has not yet committed its work.

**Wait Time:** Depends upon the amount of row lock contention in the original capture

| Parameter | Description |
|---|---|
| *wait for scn's hi 4 bytes* | High 4 bytes of the 8-byte replay SCN that the session is waiting on |
| *wait for scn's lo 4 bytes* | Low 4 bytes of the 8-byte replay SCN that the session is waiting on |

## WCR: replay lock order

A session will wait on this event during replay if it saw some lock contention during capture.

**Wait Time:** Depends upon the amount of row lock contention in the original capture

| Parameter | Description |
|---|---|
| *wait for scn's hi 4 bytes* | High 4 bytes of the 8-byte replay SCN that the session is waiting on |
| *wait for scn's lo 4 bytes* | Low 4 bytes of the 8-byte replay SCN that the session is waiting on |

## WCR: replay paused

When the user issues a DBMS_WORKLOAD_REPLAY.PAUSE_REPLAY command, all the replayed sessions are waiting on this wait event until replay is resumed.

**Parameters:** None

## WCR: Sync context busy

In the default replay mode, commits during replay are synchronized to follow the same order as capture. All commits are serialized using the Sync context. A replayed session will wait on this event if it is trying to commit at the same time as another replayed session.

**Parameters:** None

## WMON goes to sleep

WMON is the UNIX-specific Wait Monitor, that can be used to reduce the number of system calls related to setting timers for posting or waiting in Oracle. You must set an initialization parameter that enables the WMON process.

**Wait Time:** Depends on the next timeout

**Parameters:** None

## write complete waits

The session waits for a buffer to be written. The write is caused by normal aging or by a cross-instance call.

**Wait Time:** 1 second

| Parameter | Description |
|---|---|
| *file#* | The rollback segment id that contains the transaction that is being rolled back |
| *block#* | The transaction flags (options) set for the transaction that is being rolled back |
| *id* | Identifies the reason for waiting |

## writes stopped by instance recovery or database suspension

The session is blocked until the instance that started Instance Recovery is finished.

**Wait Time:** 5 seconds

| Parameter | Description |
|---|---|
| *bythread#* | The rollback segment id that contains the transaction that is being rolled back |
| *ourthread#* | The current instance thread number |

# D

# Oracle Enqueue Names

This appendix lists Oracle enqueues. **Enqueues** are shared memory structures (locks) that serialize access to database resources. They can be associated with a session or transaction. Enqueue names are displayed in the `LOCK_TYPE` column of the `DBA_LOCK` and `DBA_LOCK_INTERNAL` data dictionary views.

A **resource** uniquely identifies an object that can be locked by different sessions within an instance (local resource) or between instances (global resource). Each session that tries to lock the resource will have an **enqueue** on the resource.

---

**Note:**

The names of enqueues and their definitions may change from release to release.

---

---

**See Also:**

"DBA_LOCK_INTERNAL" and "DBA_LOCK"

---

The Oracle enqueues are:

- BL, Buffer Cache Management
- BR, Backup/Restore
- CF, Controlfile Transaction
- CI, Cross-instance Call Invocation
- CU, Bind Enqueue
- DF, Datafile
- DL, Direct Loader Index Creation
- DM, Database Mount
- DR, Distributed Recovery Process
- DW, SecureFiles
- DX, Distributed Transaction
- FP, File Object
- FS, File Set

- HW, High-Water Lock
- IN, Instance Number
- IR, Instance Recovery
- IS, Instance State
- IV, Library Cache Invalidation
- JI, Enqueue used during AJV snapshot refresh
- JQ, Job Queue
- KK, Redo Log "Kick"
- KP, contention in Oracle Data Pump startup and shutdown processes
- KO, Multiple Object Checkpoint
- L[A-P], Library Cache Lock
- LS, Log Start or Switch
- MM, Mount Definition
- MR, Media Recovery
- N[A-Z], Library Cache Pin
- PE, ALTER SYSTEM SET PARAMETER = VALUE
- PF, Password File
- PI, Parallel Slaves
- PR, Process Startup
- PS, Parallel Slave Synchronization
- Q[A-Z], Row Cache
- RO, Object Reuse
- RT, Redo Thread
- RW, Row Wait
- SC, System Change Number
- SM, SMON
- SN, Sequence Number
- SQ, Sequence Number Enqueue
- SR, Synchronized Replication
- SS, Sort Segment
- ST, Space Management Transaction
- SV, Sequence Number Value

- TA, Transaction Recovery
- TC, Thread Checkpoint
- TE, Extend Table
- TM, DML Enqueue
- TO, Temporary Table Object Enqueue
- TS, Temporary Segment (also TableSpace)
- TT, Temporary Table
- TX, Transaction
- UL, User-defined Locks
- UN, User Name
- US, Undo Segment, Serialization
- WL, Being Written Redo Log
- XA, Instance Attribute Lock
- XI, Instance Registration Lock
- ZA, Exclusive Lock When Moving Audit Table

# E

# Statistics Descriptions

This appendix describes the statistics stored in the V$SESSTAT and V$SYSSTAT dynamic performance tables. These statistics are useful in identifying and correcting performance problems.

This appendix contains the following topics:


- Displaying Statistics
- Statistics Descriptions


## Displaying Statistics

The V$SESSTAT view displays statistics on a per-session basis and is valid only for the session currently connected. When a session disconnects, all statistics for the session are updated in V$SYSSTAT. The values for the statistics are cleared until the next session uses them.

The V$STATNAME view contains all of the statistics for an Oracle release.

Many of these statistics are tied to the internal implementation of Oracle and therefore are subject to change or deletion without notice, even between patch releases. Application developers should be aware of this and write their code to tolerate missing or extra statistics.

---

**See Also:**

"V$SESSTAT", "V$STATNAME", and "V$SYSSTAT" for more information on these views

---

## Statistics Descriptions

This section describes the statistics stored in the V$SESSTAT and V$SYSSTAT views. The statistics are listed here in alphabetical order.

The CLASS column contains a number representing one or more statistics classes. The following class numbers are additive:

- 1, User
- 2, Redo
- 4, Enqueue
- 8, Cache
- 16, OS

- 32, Real Application Clusters
- 64, SQL
- 128, Debug

For example, a class value of 72 represents a statistic that relates to SQL statements and caching.

Some statistics are populated only if the `TIMED_STATISTICS` initialization parameter is set to `true`. Those statistics are flagged with a `Y` in the right-hand column.

***Table E-1 Database Statistics Descriptions***

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| application wait time | 1 | The total wait time (in centiseconds) for waits that belong to the Application wait class | |
| background checkpoints completed | 8 | Number of checkpoints completed by the background process. This statistic is incremented when the background process successfully advances the thread checkpoint. | |
| background checkpoints started | 8 | Number of checkpoints started by the background process. This statistic can be larger than "background checkpoints completed" if a new checkpoint overrides an incomplete checkpoint or if a checkpoint is currently under way. This statistic includes only checkpoints of the redo thread. It does not include:<br>• Individual file checkpoints for operations such as offline or begin backup<br>• Foreground (user-requested) checkpoints (for example, performed by `ALTER SYSTEM CHECKPOINT LOCAL` statements) | |
| background timeouts | 128 | This is a count of the times where a background process has set an alarm for itself and the alarm has timed out rather than the background process being posted by another process to do some work. | |
| branch node splits | 128 | Number of times an index branch block was split because of the insertion of an additional value | |
| buffer is not pinned count | 72 | Number of times a buffer was free when visited. Useful only for internal debugging purposes. | |
| buffer is pinned count | 72 | Number of times a buffer was pinned when visited. Useful only for internal debugging purposes. | |
| bytes received via SQL*Net from client | 1 | Total number of bytes received from the client over Oracle Net Services | |
| bytes received via SQL*Net from dblink | 1 | Total number of bytes received from a database link over Oracle Net Services | |
| bytes sent via SQL*Net to client | 1 | Total number of bytes sent to the client from the foreground processes | |
| bytes sent via SQL*Net to dblink | 1 | Total number of bytes sent over a database link | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| Cached Commit SCN referenced | 128 | Useful only for internal debugging purposes | |
| calls to get snapshot scn: kcmgss | 32 | Number of times a snapshot system change number (SCN) was allocated. The SCN is allocated at the start of a transaction. | |
| calls to kcmgas | 128 | Number of calls to routine kcmgas to get a new SCN | |
| calls to kcmgcs | 128 | Number of calls to routine kcmgcs to get a current SCN | |
| calls to kcmgrs | 128 | Number of calls to routine kcsgrs to get a recent SCN | |
| change write time | 8 | Elapsed redo write time for changes made to CURRENT blocks in 10s of milliseconds. | Y |
| cleanouts and rollbacks - consistent read gets | 128 | Number of consistent gets that require both block rollbacks and block cleanouts.<br>**See Also:** "consistent gets" | |
| cleanouts only - consistent read gets | 128 | Number of consistent gets that require only block cleanouts, no rollbacks.<br>**See Also:** "consistent gets" | |
| cluster key scan block gets | 64 | Number of blocks obtained in a cluster scan | |
| cluster key scans | 64 | Number of cluster scans that were started | |
| cluster wait time | 1 | The total wait time (in centiseconds) for waits that belong to the Cluster wait class | |
| cold recycle reads | 8 | Number of buffers that were read through the least recently used end of the recycle cache with fast aging strategy | |
| commit cleanout failures: block lost | 8 | Number of times Oracle attempted a cleanout at commit but could not find the correct block due to forced write, replacement, or switch CURRENT | |
| commit cleanout failures: buffer being written | 8 | Number of times Oracle attempted a cleanout at commit, but the buffer was currently being written | |
| commit cleanout failures: callback failure | 8 | Number of times the cleanout callback function returns FALSE | |
| commit cleanout failures: cannot pin | 8 | Total number of times a commit cleanout was performed but failed because the block could not be pinned | |
| commit cleanout failures: hot backup in progress | 8 | Number of times Oracle attempted block cleanout at commit during hot backup. The image of the block must be logged before the buffer can be made dirty. | |
| commit cleanout failures: write disabled | 8 | Number of times a cleanout block at commit was performed but the writes to the database had been temporarily disabled | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| commit cleanouts | 8 | Total number of times the cleanout block at commit function was performed | |
| commit cleanouts successfully completed | 8 | Number of times the cleanout block at commit function completed successfully | |
| commit nowait performed | 1 | The number of asynchronous commits that were actually performed. These commits did not wait for the commit redo to be flushed and be present on disk before returning. | |
| commit nowait requested | 1 | The number of no-wait commit or asynchronous commit requests that were made either using SQL or the OCI transaction control API | |
| Commit SCN cached | 128 | Number of times the system change number of a commit operation was cached | |
| commit wait/nowait performed | 1 | The number of asynchronous/synchronous commits that were actually performed | |
| commit wait/nowait requested | 1 | The number of no-wait or wait commits that were made either using SQL or the OCI transaction control API | |
| commit wait performed | 1 | The number of synchronous commits that were actually performed. These commits waited for the commit redo to be flushed and be present on disk before returning. | |
| commit wait requested | 1 | The number of waiting or synchronous commit requests that were made either using SQL or the OCI transaction control API | |
| concurrency wait time | 1 | The total wait time (in centiseconds) for waits that belong to the Concurrency wait class | |
| consistent changes | 8 | Number of times a user process has applied rollback entries to perform a consistent read on the block<br>Work loads that produce a great deal of consistent changes can consume a great deal of resources. The value of this statistic should be small in relation to the "consistent gets" statistic. | |
| consistent gets | 8 | Number of times a consistent read was requested for a block.<br>**See Also:** "consistent changes" and "session logical reads" statistics | |
| consistent gets direct | 8 | Number of times a consistent read was requested for a block bypassing the buffer cache (for example, direct load operation). This is a subset of "consistent gets" statistics value. | |
| consistent gets from cache | 8 | Number of times a consistent read was requested for a block from buffer cache. This is a subset of "consistent gets" statistics value. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| CPU used by this session | 1 | Amount of CPU time (in 10s of milliseconds) used by a session from the time a user call starts until it ends. If a user call completes within 10 milliseconds, the start and end user-call time are the same for purposes of this statistics, and 0 milliseconds are added.<br>A similar problem can exist in the reporting by the operating system, especially on systems that suffer from many context switches. | Y |
| CPU used when call started | 128 | The CPU time used when the call is started<br>**See Also:** "CPU used by this session" | Y |
| CR blocks created | 8 | Number of `CURRENT` blocks cloned to create CR (consistent read) blocks. The most common reason for cloning is that the buffer is held in a incompatible mode. | |
| current blocks converted for CR | 8 | Number `CURRENT` blocks converted to CR state | |
| cursor authentications | 128 | Number of privilege checks conducted during execution of an operation | |
| data blocks consistent reads - undo records applied | 128 | Number of undo records applied to data blocks that have been rolled back for consistent read purposes | |
| data warehousing cooling action | 8 | Number of times that cooling occurred on this instance | |
| data warehousing evicted objects | 8 | Number of times that objects got evicted by automatic big table caching on this instance | |
| data warehousing evicted objects - cooling | 8 | Number of times that objects got evicted on this instance due to a cooling action | |
| data warehousing evicted objects - replace | 8 | Number of times that objects got evicted due to caching replacement, that is, when an object is evicted because a hotter object forces it to be evicted from the cache | |
| data warehousing scanned blocks | 8 | Number of blocks scanned by automatic big table caching on this instance using parallel query | |
| data warehousing scanned blocks - disk | 8 | Number of blocks scanned by automatic big table caching on this instance by direct read from disk | |
| data warehousing scanned blocks - memory | 8 | Number of blocks scanned by automatic big table caching on this instance by cache read from memory | |
| data warehousing scanned blocks - offload | 8 | Number of blocks scanned by automatic big table caching on this instance by Exadata offloading | |
| data warehousing scanned objects | 8 | Number of times the objects in automatic big table caching are scanned using parallel query | |
| db block changes | 8 | Closely related to "consistent changes", this statistic counts the total number of changes that were part of an update or delete operation that were made to all blocks in the SGA. Such changes generate redo log entries and | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| | | hence become permanent changes to the database if the transaction is committed.<br>This approximates total database work. This statistic indicates the rate at which buffers are being dirtied (on a per-transaction or per-second basis, for example). | |
| db block gets | 8 | Number of times a `CURRENT` block was requested<br>**See Also:** "consistent gets" | |
| db block gets direct | 8 | Number of times a `CURRENT` block was requested bypassing the buffer cache (for example, a direct load operation). This is a subset of "db block gets" statistics value. | |
| db block gets from cache | 8 | Number of times a `CURRENT` block was requested from the buffer cache. This is a subset of "db block gets" statistics value. | |
| DBWR checkpoint buffers written | 8 | Number of buffers that were written for checkpoints | |
| DBWR checkpoints | 8 | Number of times the DBWR was asked to scan the cache and write all blocks marked for a checkpoint or the end of recovery. This statistic is always larger than "background checkpoints completed". | |
| DBWR lru scans | 8 | Number of times that DBWR scans the LRU queue looking for buffers to write. This count includes scans to fill a batch being written for another purpose (such as a checkpoint). | |
| DBWR revisited being-written buffer | 8 | Number of times that DBWR tried to save a buffer for writing and found that it was already in the write batch. This statistic measures the amount of "useless" work that DBWR had to do in trying to fill the batch.<br>Many sources contribute to a write batch. If the same buffer from different sources is considered for adding to the write batch, then all but the first attempt will be "useless" because the buffer is already marked as being written. | |
| DBWR transaction table writes | 8 | Number of rollback segment headers written by DBWR. This statistic indicates how many "hot" buffers were written, causing a user process to wait while the write completed. | |
| DBWR undo block writes | 8 | Number of rollback segment blocks written by DBWR | |
| DDL statements parallelized | 32 | Number of DDL statements that were executed in parallel | |
| deferred (CURRENT) block cleanout applications | 128 | Number of times cleanout records are deferred, piggyback with changes, always current get | |
| DFO trees parallelized | 32 | Number of times a serial execution plan was converted to a parallel plan | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| dirty buffers inspected | 8 | Number of dirty buffers found by the user process while it is looking for a buffer to reuse | |
| DML statements parallelized | 32 | Number of DML statements that were executed in parallel | |
| enqueue conversions | 4 | Total number of conversions of the state of table or row lock | |
| enqueue deadlocks | 4 | Total number of deadlocks between table or row locks in different sessions | |
| enqueue releases | 4 | Total number of table or row locks released | |
| enqueue requests | 4 | Total number of table or row locks acquired | |
| enqueue timeouts | 4 | Total number of table and row locks (acquired and converted) that timed out before they could complete | |
| enqueue waits | 4 | Total number of waits that occurred during an enqueue convert or get because the enqueue get was deferred | |
| exchange deadlocks | 8 | Number of times that a process detected a potential deadlock when exchanging two buffers and raised an internal, restartable error. Index scans are the only operations that perform exchanges. | |
| execute count | 64 | Total number of calls (user and recursive) that executed SQL statements | |
| fbda woken up | 128 | Number of times the flashback data archive background process was woken up to do archiving | |
| file io wait time | 1 | Total time spent in wait (in microseconds) for I/O to datafiles, excluding the service time for such I/O. This is cumulative for all I/Os for all datafiles. The service time for one I/O operation is estimated as the minimum time spent in the I/O call seen so far. This service time is subtracted from the time spent in each I/O call to get the wait time for that I/O. | |
| flash cache eviction: aged out | 8 | Flash cache buffer is aged out of the Database Smart Flash Cache | |
| flash cache eviction: buffer pinned | 8 | Database Smart Flash Cache buffer is invalidated due to object or range reuse, and so on. The Database Flash Cache Buffer was in use at the time of eviction. | |
| flash cache eviction: invalidated | 8 | Database Smart Flash Cache buffer is invalidated due to object or range reuse, and so on. The Database Smart Flash Cache buffer was not in use at the time of eviction. | |
| flash cache insert skip: corrupt | 8 | In-memory buffer was skipped for insertion into the Database Smart Flash Cache because the buffer was corrupted | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| flash cache insert skip: DBWR overloaded | 8 | In-memory buffer was skipped for insertion into the Database Smart Flash Cache because DBWR was busy writing other buffers | |
| flash cache insert skip: exists | 8 | In-memory buffer was skipped for insertion into the Database Smart Flash Cache because it was already in the flash cache | |
| flash cache insert skip: modification | 8 | In-memory buffer was skipped for insertion into the Database Smart Flash Cache because it was being modified | |
| flash cache insert skip: not current | 8 | In-memory buffer was skipped for insertion into the Database Smart Flash Cache because it was not current | |
| flash cache insert skip: not useful | 8 | In-memory buffer was skipped for insertion into the Database Smart Flash Cache because the type of buffer was not useful to keep | |
| flash cache inserts | 8 | Total number of in-memory buffers inserted into the Database Smart Flash Cache | |
| flashback log write bytes | 2 | Total size in bytes of flashback database data written by RVWR to flashback database logs | |
| flashback log writes | 2 | Total number of writes by RVWR to flashback database logs | |
| foreground propagated tracked transactions | 128 | Number of transactions modifying tables enabled for flashback data archive which were archived by a foreground process | |
| free buffer inspected | 8 | Number of buffers skipped over from the end of an LRU queue to find a reusable buffer. The difference between this statistic and "dirty buffers inspected" is the number of buffers that could not be used because they had a user, a waiter, or were being read or written, or because they were busy or needed to be written after rapid aging out. | |
| free buffer requested | 8 | Number of times a reusable buffer or a free buffer was requested to create or load a block | |
| gc read wait failures | 40 | A read wait is when a CR server waits for a disk read to complete before serving a block to another instance. This statistic displays the number of times a read wait ended in failure, that is, after waiting it was unable to serve a block. | |
| gc read wait time | 40 | A read wait is when a CR server waits for a disk read to complete before serving a block to another instance. This statistic displays the total time spent waiting for disk reads (in centiseconds). The average time waited is "gc read wait time" / "gc read waits." | |
| gc read wait timeouts | 40 | A read wait is when a CR server waits for a disk read to complete before serving a block to another instance. This statistic displays the number of times a read wait | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| | | timed out, that is, the disk read did not complete in time, so the wait was aborted. | |
| gc read waits | 40 | The number of times a CR server waited for a disk read, and then successfully served a block | |
| global cache blocks corrupt | 40 | Real Application Clusters only: Number of blocks that encountered a corruption or checksum failure during interconnect | |
| global cache convert time | 40 | Real Application Clusters only: Total time elapsed during lock converts | |
| global cache convert timeouts | 40 | Number of times lock converts in the global cache timed out | |
| global cache converts | 40 | Number of lock converts in the global cache | |
| global cache cr block log flushes | 40 | Number of log flushes of the consistent-read block | |
| global cache cr block log flush time | 40 | Total time spent by the BSP process in log flushes after sending a constructed consistent-read (CR) block. This statistic divided by "global cache cr blocks served" = log flush time per CR block. | |
| global cache cr block receive time | 40 | Total amount of time foreground processes waited for a CR block to be sent through the interconnect. This statistic divided by "global cache cr blocks received" = time waited per block. | |
| global cache cr block send time | 40 | Total time spent by the BSP process in sending constructed consistent-read (CR) blocks. This statistic divided by "global cache cr blocks served" = send time per CR block. | |
| global cache cr block serve time | 40 | Total amo<br>unt of time the BSP process took to construct consistent-read (CR) blocks. This statistic divided by "global cache cr blocks served" = construction time per CR block. | |
| global cache cr blocks received | 40 | Total number of blocks received | |
| global cache cr blocks served | 40 | Total number of blocks constructed by the BSP process | |
| global cache cr requests blocked | 40 | Number of times foreground attempt to request a cr block and failed | |
| global cache cr timeouts | 40 | Number of times a foreground process requested a consistent-read (CR) block when the request timed out | |
| global cache defers | 40 | Number of times a lock was requested and the holder of the lock deferred the release | |
| global cache freelist waits | 40 | System configured with fewer lock elements than buffers. Number of times foreground has to wait for a lock element. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| global cache get time | 40 | Total time spent waiting. This divided by global cache gets = time waited per request. | |
| global cache gets | 40 | Number of locks acquired | |
| global cache prepare failures | 40 | Number of times a failure occurred during preparation for interconnect transfer | |
| global enqueue get time | 32 | Total elapsed time in 10s of milliseconds of all synchronous and asynchronous global enqueue gets and converts | |
| global enqueue gets async | 32 | Total number of asynchronous global enqueue gets and converts | |
| global enqueue gets sync | 32 | Total number of synchronous global enqueue gets and converts | |
| global enqueue releases | 32 | Total number of synchronous global enqueue releases | |
| global enqueue CPU used by this session | 32 | Amount of CPU time (in 10s of milliseconds) used by synchronous and asynchronous global enqueue activity in a session from the time a user call starts until it ends. If a user call completes within 10 milliseconds, the start and end user-call time are the same for purposes of this statistics, and 0 milliseconds are added. | |
| hot buffers moved to head of LRU | 8 | When a hot buffer reaches the tail of its replacement list, Oracle moves it back to the head of the list to keep it from being reused. This statistic counts such moves. | |
| in call idle wait time | 1 | The total wait time (in microseconds) for waits that belong to the Idle wait class.<br>**See Also:** "non-idle wait count" and "non-idle wait time" | |
| immediate (CR) block cleanout applications | 128 | Number of times cleanout records are applied immediately during consistent-read requests | |
| immediate (CURRENT) block cleanout applications | 128 | Number of times cleanout records are applied immediately during current gets. Compare this statistic with "deferred (CURRENT) block cleanout applications" | |
| index fast full scans (direct read) | 64 | Number of fast full scans initiated using direct read | |
| index fast full scans (full) | 64 | Number of fast full scans initiated for full segments | |
| index fast full scans (rowid ranges) | 64 | Number of fast full scans initiated with rowid endpoints specified | |
| instance recovery database freeze count | 32 | Number of times the database is frozen during instance recovery | |
| kcmccs called get current scn | 32 | Number of times the kernel got the CURRENT SCN when there was a need to casually confirm the SCN | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| kcmgss read scn without going to DLM | 32 | Number of times the kernel got a snapshot SCN without going to the distributed lock manager (DLM) | |
| kcmgss waited for batching | 32 | Number of times a database process is blocked waiting for a snapshot SCN | |
| large tracked transactions | 128 | For tables tracked by flashback data archive, the number of transactions modifying rows in those tables which are large in terms of size or number of changes | |
| leaf node splits | 128 | Number of times an index leaf node was split because of the insertion of an additional value | |
| lob reads | 8 | Number of LOB API read operations performed in the session/system. A single LOB API read may correspond to multiple physical/logical disk block reads. | |
| lob writes | 8 | Number of LOB API write operations performed in the session/system. A single LOB API write may correspond to multiple physical/logical disk block writes. | |
| lob writes unaligned | 8 | Number of LOB API write operations whose start offset or buffer size is not aligned to the internal chunk size of the LOB. Writes aligned to chunk boundaries are the most efficient write operations. The internal chunk size of a LOB is available through the LOB API (for example, DBMS_LOB.GETCHUNKSIZE()). | |
| logons cumulative | 1 | Total number of logons since the instance started. Useful only in V$SYSSTAT. It gives an instance overview of all processes that logged on. | |
| logons current | 1 | Total number of current logons. Useful only in V $SYSSTAT. | |
| messages received | 128 | Number of messages sent and received between background processes | |
| messages sent | 128 | Number of messages sent and received between background processes | |
| native hash arithmetic execute | 64 | Number of hash operations performed using native arithmetic rather than Oracle NUMBERs | |
| native hash arithmetic fail | 64 | Number of has operations performed using native arithmetic that failed, requiring the hash operation to be performed with Oracle NUMBERs | |
| next scns gotten without going to DLM | 32 | Number of system change numbers obtained without going to the distributed lock manager or server | |
| no buffer to keep pinned count | 72 | Number of times a visit to a buffer attempted, but the buffer was not found where expected. Like "buffer is not pinned count" and "buffer is pinned count", this statistic is useful only for internal debugging purposes. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| no work - consistent read gets | 128 | Number consistent gets that require neither block cleanouts nor rollbacks.<br>**See Also:** "consistent gets" | |
| non-idle wait count | 1 | The total number of waits performed with wait events that were not part of the Idle wait class.<br>**See Also:** "in call idle wait time" and "non-idle wait time" | |
| non-idle wait time | 1 | The total wait time (in microseconds) for waits that do not belong to the Idle wait class.<br>**See Also:** "in call idle wait time" and "non-idle wait count" | |
| OLAP Paging Manager Cache Hit | 64 | The number of times a requested page is found in the OLAP page pool. Use this statistic in conjunction with OLAP Paging Manager Cache Miss to determine the OLAP page pool efficiency ratio. | |
| OLAP Paging Manager Cache Miss | 64 | The number of times a requested page is not found in the OLAP page pool. Use this statistic in conjunction with OLAP Paging Manager Cache Hit to determine the OLAP page pool efficiency ratio. | |
| OLAP Paging Manager New Page | 64 | The number of newly-created pages in the OLAP page pool that have not yet been written to the workspace LOB | |
| OLAP Paging Manager Cache Write | 64 | The number of times the OLAP paging manager writes to a page in the OLAP page pool. | |
| OLAP Session Cache Hit | 64 | The number of times the requested, dynamically-aggregated value of an AW object, was found in the OLAP session cache. | |
| OLAP Session Cache Miss | 64 | The number of times the requested, dynamically-aggregated value of an AW object, was not found in the OLAP session cache. | |
| OLAP Aggregate Function Calc | 64 | The number of times the `AGGREGATE` function computes a parent value based on the values of its children. | |
| OLAP Aggregate Function Precompute | 64 | The number of times the `AGGREGATE` function is to compute a value and finds it precomputed in the cube. | |
| OLAP Aggregate Function Logical NA | 64 | The number of times an `AGGREGATE` function evaluates to a logical NA value. This could be because the `AGGINDEX` is on and the composite tuple does not exist. | |
| OLAP Paging Manager Pool Size | 64 | Size, in bytes, of the OLAP page pool allocated to a session and the sum of all OLAP page pools in the system. | |
| OLAP Import Rows Pushed | 64 | The number of OLAP import rows pushed. This statistic refers to the number of rows encountered from a source cursor and is useful during cube build operations. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| OLAP Import Rows Loaded | 64 | The number of OLAP import rows loaded. This statistic provides the number of rows of the source cursor that are actually loaded into an Analytic Workspace (AW).<br>The difference between the OLAP Import Rows Pushed and OLAP Import Rows Loaded provides the number of rejected rows. | |
| OLAP Row Source Rows Processed | 64 | The number of rows processed by the OLAP row source | . |
| OLAP Engine Calls | 64 | The total number of OLAP transactions executed within the session. This value provides a general indication of the level of OLAP activity in the session. | |
| OLAP Temp Segments | 64 | The number of OLAP pages stored in temporary segments for analytic workspaces | |
| OLAP Temp Segments Read | 64 | The number of times data was read from a temporary segment and not from the OLAP page pool | |
| OLAP Perm LOB Read | 64 | The number of times data was read from the table where the AW is stored. These are permanent LOB reads. | |
| OLAP Paging Manager Cache Changed Page | 64 | The number of times the OLAP page pool is changed for any attached AW. | |
| OLAP Fast Limit | 64 | The number of times an OLAP table function issues a fast limit | |
| OLAP GID Limit | 64 | The number of times an OLAP table function issues a Cube Grouping ID (CGID) limit. Typically, this type of limit occurs for query rewrite transformations that resolve to a cube organized materialized view. | |
| OLAP INHIER Limit | 64 | The number of times an OLAP table function issues an in-hierarchy limit. This type of limit can occur when you use cube dimension hierarchy views. | |
| OLAP Full Limit | 64 | The number of times an OLAP table function issues a full limit | |
| OLAP Custom Member Limit | 64 | The number of times an OLAP table function issues a custom member limit | |
| OLAP Row Id Limit | 64 | The number of times an OLAP table function issues a row Id limit. | |
| OLAP Limit Time | 64 | The total time taken by all the OLAP Limit operations that were performed during the last call to the OLAP table function | |
| OLAP Row Load Time | 64 | The total time spent loading rows into an AW during cube build and OLAP SQL import operations.<br>Use this statistic along with the OLAP engine elapsed time to measure time spent running OLAP engine routines that involve loading data into AWs from a SQL source. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| | | This statistic has the following levels of precision:<br>• Low precision timer<br>This captures the elapsed time of the entire fetch phase of the SQL cursor that is being loaded into AWs. It includes the SQL execution time that occurs during a fetch operation from a source cursor and time taken by the OLAP engine to populate AWs.<br>• High precision timer<br>This captures the elapsed time, excluding the SQL processing of the cursor being loaded. It records the time spent in the OLAP engine only.<br>• Default timer precision:<br>This is based on the `STATISTIC_LEVEL` parameter. If the low precision is used, then `STATISTICS_LEVEL` is TYPICAL. The high precision timer is used when `STATISTIC_LEVEL` is set to ALL. No timing is captured when `STATISTICS_LEVEL` is BASIC. | |
| OLAP Unique Key Attribute Limit | 64 | The number of times an OLAP table function issues a unique key attribute limit | |
| opened cursors cumulative | 1 | In V$SYSSTAT: Total number of cursors opened since the instance started.<br>In V$SESSTAT: Total number of cursors opened since the start of the session. | |
| opened cursors current | 1 | Total number of current open cursors | |
| opens of replaced files | 8 | Total number of files that had to be reopened because they were no longer in the process file cache | |
| opens requiring cache replacement | 8 | Total number of file opens that caused a current file in the process file cache to be closed | |
| OS All other sleep time | 16 | Time spent sleeping for reasons other than misses in the data segment (see "OS Data page fault sleep time"), kernel page faults (see "OS Kernel page fault sleep time"), or misses in the text segment (see "OS Text page fault sleep time"). An example of such a reason is expiration of quanta. | |
| OS Chars read and written | 16 | Number of bytes read and written | |
| OS CPU Qt wait time | 1 | The time a session spends on the CPU run queue (in microseconds), waiting to get the CPU to run | |
| OS Data page fault sleep time | 16 | Time spent sleeping due to misses in the data segment | |
| OS Input blocks | 16 | Number of read I/Os | |
| OS Involuntary context switches | 16 | Number of context switches that were enforced by the operating system | |
| OS Kernel page fault sleep time | 16 | Time spent sleeping due to operating system kernel page faults | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| OS Major page faults | 16 | Number of page faults that resulted in I/O | |
| OS Messages received | 16 | Number of messages received | |
| OS Messages sent | 16 | Number of messages sent | |
| OS Minor page faults | 16 | Number of page faults that did not result in an actual I/O | |
| OS Other system trap CPU time | 16 | Total amount of time to process system traps (as distinct from system calls) | |
| OS Output blocks | 16 | Number of write I/Os | |
| OS Process heap size | 16 | Size of area in memory allocated by the process. Typically this represents memory obtained by way of malloc(). | |
| OS Process stack size | 16 | Size of the process stack segment | |
| OS Signals received | 16 | Number of signals received | |
| OS Swaps | 16 | Number of swap pages | |
| OS System call CPU time | 16 | Total amount of time spent executing in system mode | |
| OS System calls | 16 | Number of system calls | |
| OS Text page fault sleep time | 16 | Time spent sleeping due to misses in the text segment | |
| OS User level CPU time | 16 | Total amount of time spent executing in user mode | |
| OS Voluntary context switches | 16 | Number of voluntary context switches (for example, when a process gives up the CPU by a SLEEP() system call) | |
| Parallel operations downgraded 1 to 25 pct | 32 | Number of times parallel execution was requested and the degree of parallelism was reduced because of insufficient parallel execution servers | |
| Parallel operations downgraded 25 to 50 pct | 32 | Number of times parallel execution was requested and the degree of parallelism was reduced because of insufficient parallel execution servers | |
| Parallel operations downgraded 50 to 75 pct | 32 | Number of times parallel execution was requested and the degree of parallelism was reduced because of insufficient parallel execution servers | |
| Parallel operations downgraded 75 to 99 pct | 32 | Number of times parallel execution was requested and the degree of parallelism was reduced because of insufficient parallel execution servers | |
| Parallel operations downgraded to serial | 32 | Number of times parallel execution was requested but execution was serial because of insufficient parallel execution servers | |
| Parallel operations not downgraded | 32 | Number of times parallel execution was executed at the requested degree of parallelism | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| parse count (hard) | 64 | Total number of parse calls (real parses). A hard parse is a very expensive operation in terms of memory use, because it requires Oracle to allocate a workheap and other memory structures and then build a parse tree. | |
| parse count (describe) | 64 | Total number of parse calls on a describe cursor. This operation is a less expensive than a hard parse and more expensive than a soft parse. | |
| parse count (total) | 64 | Total number of parse calls (hard, soft, and describe). A soft parse is a check on an object already in the shared pool, to verify that the permissions on the underlying object have not changed. | |
| parse time cpu | 64 | Total CPU time used for parsing (hard and soft) in 10s of milliseconds | Y |
| parse time elapsed | 64 | Total elapsed time for parsing, in 10s of milliseconds. Subtract "parse time cpu" from the this statistic to determine the total waiting time for parse resources. | Y |
| physical read bytes | 8 | Total size in bytes of all disk reads by application activity (and not other instance activity) only. | |
| physical read flash cache hits | 8 | Total number of reads from flash cache instead of disk | |
| physical read IO requests | 8 | Number of read requests for application activity (mainly buffer cache and direct load operation) which read one or more database blocks per request. This is a subset of "physical read total IO requests" statistic. | |
| physical read requests optimized | 8 | Number of read requests that read one or more database blocks from the Database Smart Flash Cache or the Exadata Smart Flash Cache. | |
| physical read total bytes | 8 | Total size in bytes of disk reads by all database instance activity including application reads, backup and recovery, and other utilities. The difference between this value and "physical read bytes" gives the total read size in bytes by non-application workload. | |
| physical read total IO requests | 8 | Number of read requests which read one or more database blocks for all instance activity including application, backup and recovery, and other utilities. The difference between this value and "physical read total multi block requests" gives the total number of single block read requests. | |
| physical read total multi block requests | 8 | Total number of Oracle instance read requests which read in two or more database blocks per request for all instance activity including application, backup and recovery, and other utilities. | |
| physical reads | 8 | Total number of data blocks read from disk. This value can be greater than the value of "physical reads direct" plus "physical reads cache" as reads into process private buffers also included in this statistic. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| physical reads cache | 8 | Total number of data blocks read from disk into the buffer cache. This is a subset of "physical reads" statistic. | |
| physical reads cache prefetch | 8 | Number of contiguous and noncontiguous blocks that were prefetched. | |
| physical reads direct | 8 | Number of reads directly from disk, bypassing the buffer cache. For example, in high bandwidth, data-intensive operations such as parallel query, reads of disk blocks bypass the buffer cache to maximize transfer rates and to prevent the premature aging of shared data blocks resident in the buffer cache. | |
| physical reads direct (lob) | 8 | Number of buffers that were read directly for LOBs | |
| physical reads direct temporary tablespace | 8 | Number of buffers that were read directly from temporary tablespaces | |
| physical reads for flashback new | 8 | Number of blocks read for newing (that is, preparing a data block for a completely new change) blocks while flashback database is enabled | |
| physical reads prefetch warmup | 8 | Number of data blocks that were read from the disk during the automatic prewarming of the buffer cache. | |
| physical write bytes | 8 | Total size in bytes of all disk writes from the database application activity (and not other kinds of instance activity). | |
| physical write IO requests | 8 | Number of write requests for application activity (mainly buffer cache and direct load operation) which wrote one or more database blocks per request. | |
| physical write total bytes | 8 | Total size in bytes of all disk writes for the database instance including application activity, backup and recovery, and other utilities. The difference between this value and "physical write bytes" gives the total write size in bytes by non-application workload. | |
| physical write total IO requests | 8 | Number of write requests which wrote one or more database blocks from all instance activity including application activity, backup and recovery, and other utilities. The difference between this stat and "physical write total multi block requests" gives the number of single block write requests. | |
| physical write total multi block requests | 8 | Total number of Oracle instance write requests which wrote two or more blocks per request to the disk for all instance activity including application activity, recovery and backup, and other utilities. | |
| physical writes | 8 | Total number of data blocks written to disk. This statistics value equals the sum of "physical writes direct" and "physical writes from cache" values. | |
| physical writes direct | 8 | Number of writes directly to disk, bypassing the buffer cache (as in a direct load operation) | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| physical writes direct (lob) | 8 | Number of buffers that were directly written for LOBs | |
| physical writes direct temporary tablespace | 8 | Number of buffers that were directly written for temporary tablespaces | |
| physical writes from cache | 8 | Total number of data blocks written to disk from the buffer cache. This is a subset of "physical writes" statistic. | |
| physical writes non checkpoint | 8 | Number of times a buffer is written for reasons other than advancement of the checkpoint. Used as a metric for determining the I/O overhead imposed by setting the FAST_START_IO_TARGET parameter to limit recovery I/Os. (Note that FAST_START_IO_TARGET is a deprecated parameter.) Essentially this statistic measures the number of writes that would have occurred had there been no checkpointing. Subtracting this value from "physical writes" gives the extra I/O for checkpointing. | |
| pinned buffers inspected | 8 | Number of times a user process, when scanning the tail of the replacement list looking for a buffer to reuse, encountered a cold buffer that was pinned or had a waiter that was about to pin it. This occurrence is uncommon, because a cold buffer should not be pinned very often. | |
| prefetched blocks aged out before use | 8 | Number of contiguous and noncontiguous blocks that were prefetched but aged out before use | |
| process last non-idle time | 128 | The last time this process executed | Y |
| PX local messages recv'd | 32 | Number of local messages received for parallel execution within the instance local to the current session | |
| PX local messages sent | 32 | Number of local messages sent for parallel execution within the instance local to the current session | |
| PX remote messages recv'd | 32 | Number of remote messages received for parallel execution within the instance local to the current session | |
| PX remote messages sent | 32 | Number of remote messages sent for parallel execution within the instance local to the current session | |
| queries parallelized | 32 | Number of SELECT statements executed in parallel | |
| recovery array read time | 8 | Elapsed time of I/O during recovery | |
| recovery array reads | 8 | Number of reads performed during recovery | |
| recovery blocks read | 8 | Number of blocks read during recovery | |
| recovery blocks read for lost write detection | 8 | Number of blocks read for lost write checks during recovery. | |
| recovery blocks skipped lost write checks | 8 | Number of Block Read Records that skipped the lost write check during recovery. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| recursive calls | 1 | Number of recursive calls generated at both the user and system level. Oracle maintains tables used for internal processing. When Oracle needs to make a change to these tables, it internally generates an internal SQL statement, which in turn generates a recursive call. | |
| recursive cpu usage | 1 | Total CPU time used by non-user calls (recursive calls). Subtract this value from "CPU used by this session" to determine how much CPU time was used by the user calls. | |
| redo blocks checksummed by FG (exclusive) | 2 | Number of exclusive redo blocks that were checksummed by the generating foreground processes. An exclusive redo block is the one whose entire redo content belongs to a single redo entry. | |
| redo blocks checksummed by LGWR | 2 | Number of redo blocks that were checksummed by the LGWR. | |
| redo blocks written | 2 | Total number of redo blocks written. This statistic divided by "redo writes" equals number of blocks per write. | |
| redo buffer allocation retries | 2 | Total number of retries necessary to allocate space in the redo buffer. Retries are needed either because the redo writer has fallen behind or because an event such as a log switch is occurring. | |
| redo entries | 2 | Number of times a redo entry is copied into the redo log buffer | |
| redo entries for lost write detection | 2 | Number of times a Block Read Record is copied into the log buffer. | |
| redo log space requests | 2 | Number of times the active log file is full and Oracle must wait for disk space to be allocated for the redo log entries. Such space is created by performing a log switch.<br>Log files that are small in relation to the size of the SGA or the commit rate of the work load can cause problems. When the log switch occurs, Oracle must ensure that all committed dirty buffers are written to disk before switching to a new log file. If you have a large SGA full of dirty buffers and small redo log files, a log switch must wait for DBWR to write dirty buffers to disk before continuing.<br>Also examine the **log file space** and **log file space switch** wait events in `V$SESSION_WAIT` | |
| redo log space wait time | 2 | Total time waited in centiseconds for available space in the redo log buffer. See also"redo log space requests" | Y |
| redo ordering marks | 2 | Number of times that a system change number was allocated to force a redo record to have a higher SCN than a record generated in another thread using the same block | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| redo size | 2 | Total amount of redo generated in bytes | |
| redo size for lost write detection | 2 | Total amount of Block Read Records generated in bytes. | |
| redo synch time | 8 | Elapsed time of all "redo synch writes" calls in 10s of milliseconds | Y |
| redo synch writes | 8 | Number of times the redo is forced to disk, usually for a transaction commit. The log buffer is a circular buffer that LGWR periodically flushes. Usually, redo that is generated and copied into the log buffer need not be flushed out to disk immediately. | |
| redo wastage | 2 | Number of bytes wasted because redo blocks needed to be written before they are completely full. Early writing may be needed to commit transactions, to be able to write a database buffer, or to switch logs. | |
| redo write broadcast ack count | 2 | Number of times a commit broadcast acknowledgment has not been received by the time when the corresponding log write is completed. This is only for Oracle RAC. | |
| redo write broadcast ack time | 2 | Total amount of the latency associated with broadcast on commit beyond the latency of the log write (in microseconds). This is only for Oracle RAC. | Y |
| redo write time | 2 | Total elapsed time of the write from the redo log buffer to the current redo log file in 10s of milliseconds | Y |
| redo writes | 2 | Total number of writes by LGWR to the redo log files. "redo blocks written" divided by this statistic equals the number of blocks per write | |
| remote instance undo block writes | 40 | Number of times this instance wrote a rollback segment so that another instance could read it | |
| remote instance undo header writes | 40 | Number of times this instance wrote a undo header block so that another instance could read it | |
| rollback changes - undo records applied | 128 | Number of undo records applied to user-requested rollback changes (not consistent-read rollbacks) | |
| rollbacks only - consistent read gets | 128 | Number of consistent gets that require only block rollbacks, no block cleanouts.<br>**See Also:** "consistent gets" | |
| rows fetched via callback | 64 | Rows fetched via callback. Useful primarily for internal debugging purposes. | |
| scheduler wait time | 1 | The total wait time (in microseconds) for waits that belong to the Scheduler wait class | |
| SCN increments due to another database | 128 | SCN increments due to communication with another database | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| serializable aborts | 1 | Number of times a SQL statement in a serializable isolation level had to abort | |
| session connect time | 1 | The connect time for the session in 10s of milliseconds. This value is useful only in V$SESSTAT. It is the wall clock time since the logon to this session occurred. | Y |
| session cursor cache count | 64 | Total number of cursors cached. This statistic is incremented only if SESSION_CACHED_CURSORS > 0. This statistic is the most useful in V$SESSTAT. If the value for this statistic in V$SESSTAT is close to the setting of the SESSION_CACHED_CURSORS parameter, the value of the parameter should be increased. | |
| session cursor cache hits | 64 | Number of hits in the session cursor cache. A hit means that the SQL (including recursive SQL) or PL/SQL statement did not have to be reparsed. Subtract this statistic from "parse count (total)" to determine the real number of parses that occurred. | |
| session logical reads | 1 | The sum of "db block gets" plus "consistent gets". This includes logical reads of database blocks from either the buffer cache or process private memory. | |
| session pga memory | 1 | Current PGA size for the session. Useful only in V$SESSTAT; it has no meaning in V$SYSSTAT. | |
| session pga memory max | 1 | Peak PGA size for the session. Useful only in V$SESSTAT; it has no meaning in V$SYSSTAT. | |
| session stored procedure space | 1 | Amount of memory this session is using for stored procedures | |
| session uga memory | 1 | Current UGA size for the session. Useful only in V$SESSTAT; it has no meaning in V$SYSSTAT. | |
| session uga memory max | 1 | Peak UGA size for a session. Useful only in V$SESSTAT; it has no meaning in V$SYSSTAT. | |
| shared hash latch upgrades - no wait | 8 | A shared hash latch upgrade is when a hash latch is upgraded from shared mode to exclusive mode. This statistic displays the number of times the upgrade completed immediately. | |
| shared hash latch upgrades - wait | 8 | A shared hash latch upgrade is when a hash latch is upgraded from shared mode to exclusive mode. This statistics displays the number of times the upgrade did not complete immediately. | |
| shared io pool buffer get failure | 128 | Number of unsuccessful buffer gets from the shared I/O pool from instance startup time. | |
| shared io pool buffer get success | 128 | Number of successful buffer gets from the shared I/O pool from instance startup time. | |
| slave propagated tracked transactions | 128 | Number of transactions modifying tables enabled for flashback data archive which were archived by a slave process | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| sorts (disk) | 64 | Number of sort operations that required at least one disk write<br>Sorts that require I/O to disk are quite resource intensive. Try increasing the size of the initialization parameter `SORT_AREA_SIZE`. For more information, see "SORT_AREA_SIZE". | |
| sorts (memory) | 64 | Number of sort operations that were performed completely in memory and did not require any disk writes<br>You cannot do much better than memory sorts, except maybe no sorts at all. Sorting is usually caused by selection criteria specifications within table join SQL operations. | |
| sorts (rows) | 64 | Total number of rows sorted | |
| SQL*Net roundtrips to/from client | 1 | Total number of Oracle Net Services messages sent to and received from the client | |
| SQL*Net roundtrips to/from dblink | 1 | Total number of Oracle Net Services messages sent over and received from a database link | |
| summed dirty queue length | 8 | The sum of the dirty LRU queue length after every write request. Divide by **write requests** to get the average queue length after write completion. | |
| switch current to new buffer | 8 | Number of times the CURRENT block moved to a different buffer, leaving a CR block in the original buffer | |
| table fetch by rowid | 64 | Number of rows that are fetched using a ROWID (usually recovered from an index)<br>This occurrence of table scans usually indicates either non-optimal queries or tables without indexes. Therefore, this statistic should increase as you optimize queries and provide indexes in the application. | |
| table fetch continued row | 64 | Number of times a chained or migrated row is encountered during a fetch<br>Retrieving rows that span more than one block increases the logical I/O by a factor that corresponds to the number of blocks than need to be accessed. Exporting and re-importing may eliminate this problem. Evaluate the settings for the storage parameters PCTFREE and PCTUSED. This problem cannot be fixed if rows are larger than database blocks (for example, if the `LONG` data type is used and the rows are extremely large). | |
| table scan blocks gotten | 64 | During scanning operations, each row is retrieved sequentially by Oracle. This statistic counts the number of blocks encountered during the scan.<br>This statistic tells you the number of database blocks that you had to get from the buffer cache for the purpose of scanning. Compare this value with the value of "consistent gets" to determine how much of | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| | | the consistent read activity can be attributed to scanning. | |
| table scan rows gotten | 64 | Number of rows that are processed during scanning operations | |
| table scans (cache partitions) | 64 | Number of range scans performed on tables that have the CACHE option enabled | |
| table scans (direct read) | 64 | Number of table scans performed with direct read (bypassing the buffer cache) | |
| table scans (long tables) | 64 | Long (or conversely short) tables can be defined as tables that do not meet the short table criteria as described in table scans (short tables) | |
| table scans (rowid ranges) | 64 | During parallel query, the number of table scans conducted with specified ROWID ranges | |
| table scans (short tables) | 64 | Long (or conversely short) tables can be defined by optimizer hints coming down into the row source access layer of Oracle. The table must have the CACHE option set. | |
| total file opens | 8 | Total number of file opens performed by the instance. Each process needs several files (control file, log file, database file) in order to work against the database. | |
| tracked rows | 128 | Number of rows modified in tables enabled for flashback data archive | |
| tracked transactions | 128 | Number of transactions which modified a table enabled for flashback data archive | |
| transaction lock background get time | 128 | Useful only for internal debugging purposes | |
| transaction lock background gets | 128 | Useful only for internal debugging purposes | |
| transaction lock foreground requests | 128 | Useful only for internal debugging purposes | |
| transaction lock foreground wait time | 128 | Useful only for internal debugging purposes | |
| transaction rollbacks | 128 | Number of transactions being successfully rolled back | |
| transaction tables consistent read rollbacks | 128 | Number of times rollback segment headers are rolled back to create consistent read blocks | |
| transaction tables consistent reads - undo records applied | 128 | Number of undo records applied to transaction tables that have been rolled back for consistent read purposes | |
| Unnecessary process cleanup for SCN batching | 32 | Total number of times that the process cleanup was performed unnecessarily because the session or process did not get the next batched SCN. The next batched SCN went to another session instead. | |

| Name | Class | Description | TIMED_STATISTICS |
|---|---|---|---|
| user calls | 1 | Number of user calls such as login, parse, fetch, or execute<br>When determining activity, the ratio of user calls to RPI calls, give you an indication of how much internal work gets generated because of the type of requests the user is sending to Oracle. | |
| user commits | 1 | Number of user commits. When a user commits a transaction, the redo generated that reflects the changes made to database blocks must be written to disk. Commits often represent the closest thing to a user transaction rate. | |
| user I/O wait time | 1 | The total wait time (in centiseconds) for waits that belong to the User I/O wait class | |
| user rollbacks | 1 | Number of times users manually issue the `ROLLBACK` statement or an error occurs during a user's transactions | |
| very large tracked transactions | 128 | For tables tracked by flashback data archive, number of transactions modifying those tables which are very large in terms of size or number of changes | |
| write clones created in background | 8 | Number of times a background or foreground process clones a `CURRENT` buffer that is being written. The clone becomes the new, accessible `CURRENT` buffer, leaving the original buffer (now the clone) to complete writing. | |
| write clones created in foreground | 8 | Number of times a background or foreground process clones a `CURRENT` buffer that is being written. The clone becomes the new, accessible `CURRENT` buffer, leaving the original buffer (now the clone) to complete writing. | |

# F

# Background Processes

Table F-1 describes Oracle Database background processes. In this context, a **background process** is defined as any process that is listed in `V$PROCESS` and has a non-null value in the `PNAME` column.

The External Properties column lists the type of instance in which the process runs. If the process is specific to a particular feature, then the column names the feature.

---

**Note:**

When the `THREADED_EXECUTION` initialization parameter is set to `TRUE` on Linux and UNIX, the `DBW`, `PMON`, `PSP`, and `VKTM` background processes run as operating system processes, and the other background processes run as operating system threads.

See "THREADED_EXECUTION" for more information about the `THREADED_EXECUTION` initialization parameter.

---

***Table F-1 Background Processes***

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| ABMR | Auto BMR Background Process | Coordinates execution of tasks such as filtering duplicate block media recovery requests and performing flood control | When a process submits a block media recovery request to ABMR, it dynamically spawns slave processes (BMR*n*) to perform the recovery. ABMR and BMR*n* terminate after being idle for a long time.<br>**See Also:** *Oracle Database Backup and Recovery User's Guide* | Database instances |
| ACFS | ASM Cluster File System CSS Process | Tracks the cluster membership in CSS and informs the file system driver of membership changes | ACFS delivers CSS membership changes to the Oracle cluster file system. These membership changes are required for the file system to maintain file system consistency within the cluster. | Oracle ASM instances, Oracle RAC |
| ACMS | Atomic Control File to Memory Service Process | Coordinates consistent updates to a control file resource with its SGA counterpart on all instances in an Oracle RAC environment | The ACMS process works with a coordinating caller to ensure that an operation is executed on every instance in Oracle RAC despite failures. ACMS is the process in which a distributed operation is called. As a result, this process can exhibit a variety of behaviors. In general, ACMS is limited to small, nonblocking state changes for a limited set of cross-instance operations. | Database instances, Oracle RAC |
| AP*nn* | Database Apply | Obtains transactions from the reader | The coordinator process name is AP*nn*, where *nn* can include letters and numbers. | Database instances, |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | Process Coordinator Process | server and passes them to apply servers | For more information about the coordinator process, see `V$STREAMS_APPLY_COORDINATOR` for Oracle Streams, `V$XSTREAM_APPLY_COORDINATOR` for XStream, and `V$GG_APPLY_COORDINATOR` for Oracle GoldenGate.<br>**See Also:** *Oracle Streams Concepts and Administration* and *Oracle Database XStream Guide* | Logical Standby, Streams Apply, XStream Inbound servers, XStream Outbound servers, GoldenGate Integrated Replicat |
| AQPC | AQ Process Coordinator | Per instance AQ global coordinator | AQPC is responsible for performing administrative tasks for AQ Master Class Processes including commands like starting, stopping, and other administrative tasks. This process is automatically started on instance startup. | Database instances Advanced Queueing |
| ARB*n* | ASM Rebalance Process | Rebalances data extents within an ASM disk group | Possible processes are ARB0-ARB9 and ARBA. | Oracle ASM instances |
| ARC*n* | Archiver Process | Copies the redo log files to archival storage when they are full or an online redo log switch occurs | ARC*n* processes exist only when the database is in `ARCHIVELOG` mode and automatic archiving is enabled, in which case ARC*n* automatically archives online redo log files. LGWR cannot reuse and overwrite an online redo log group until it has been archived.<br>The database starts multiple archiver processes as needed to ensure that the archiving of filled online redo logs does not fall behind. Possible processes include ARC0-ARC9 and ARCa-ARCt.<br>The `LOG_ARCHIVE_MAX_PROCESSES` initialization parameter specifies the number of ARC*n* processes that the database initially invokes.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide* | Database instances |
| ARS*n* | ASM Recovery Slave Process | Recovers ASM transactional operations | The ASM RBAL background process coordinates and spawns one or more of these slave processes to recover aborted ASM transactional operations. These processes run only in the Oracle ASM instance.<br>Possible processes are ARS0-ARS9. | Oracle ASM instances |
| ASMB | ASM Background Process | Communicates with the ASM instance, managing storage and providing statistics | ASMB runs in Oracle ASM instances when the ASMCMD `cp` command runs or when the database instance first starts if the server parameter file is stored in Oracle ASM. ASMB also runs with Oracle Cluster Registry on Oracle ASM. | Database instances, Oracle ASM instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| AS*nn* | Database Apply Reader or Apply Server | • Computes dependencies between logical change records (LCRs) and assembles messages into transactions (Reader Server)<br>• Applies LCRs to database objects or passes LCRs and user messages to their appropriate apply handlers (Apply Server) | When the reader server finishes computing dependencies between LCRs and assembling transactions, it returns the assembled transactions to the coordinator process. Query `V$STREAMS_APPLY_READER`, `V$XSTREAM_APPLY_READER`, and `V$GG_APPLY_READER` for information about the reader server background process.<br>An apply server receives the transactions from the coordinator background process, and either applies database changes in LCRs or sends LCRs or messages to apply handlers. Apply servers can also enqueue a queue. If an apply server encounters an error, then it then tries to resolve the error with a user-specified conflict handler or error handler. If an apply server cannot resolve an error, then it rolls back the transaction and places the entire transaction, including all of its messages, in the error queue. When an apply server commits a completed transaction, this transaction has been applied. When an apply server places a transaction in the error queue and commits, this transaction also has been applied. Query `V$STREAMS_APPLY_SERVER` for information about the apply server background process. For XStream Inbound servers, query `V$XSTREAM_APPLY_SERVER`. For GoldenGate Integrated Replicat, query `V$GG_APPLY_SERVER`.<br>The coordinator process name is AS*nn*, where *nn* can include letters and numbers. | Database instances, XStream Outbound servers, XStream Inbound servers, GoldenGate Integrated Replicat |
| BMR*n* | Automatic Block Media Recovery Slave Pool Process | Fetches blocks from a real-time readable standby database | When a process submits a block media recovery request to ABMR, it dynamically spawns slave processes (BMR*n*) to perform the recovery. BMR*n* processes fetch blocks from a real-time readable standby database. ABMR and BMR*n* terminate after being idle for a long time.<br>**See Also:** *Oracle Database Backup and Recovery User's Guide* | Database instances |
| B*nnn* | ASM Blocking Slave Process for GMON | Performs maintenance actions on Oracle ASM disk groups | B*nnn* performs actions that require waiting for resources on behalf of GMON. GMON must be highly available and cannot wait.<br>A B*nnn* slave is spawned when a disk is taken offline in an Oracle ASM disk group. Offline timer processing and drop of the disk are performed in this slave. Up to five process (B000 to B004) can exist depending on the load. | Oracle ASM instances |
| BW*nn* | Database Writer Process | Writes modified blocks from the database buffer cache to the data files | See the **Long Description** for the DBW*n* process in this table for more information about the BW*nn* process. | Database instances |
| CJQ0 | Job Queue Coordinator Process | Selects jobs that need to be run from the data dictionary and | CJQ0 is automatically started and stopped as needed by Oracle Scheduler. | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | spawns job queue slave processes (J*nnn*) to run the jobs | The `JOB_QUEUE_PROCESSES` initialization parameter specifies the maximum number of processes that can be created for the execution of jobs. CJQ0 starts only as many job queue processes as required by the number of jobs to run and available resources.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide* | |
| CKPT | Checkpoint Process | Signals DBW*n* at checkpoints and updates all the data files and control files of the database to indicate the most recent checkpoint | At specific times CKPT starts a checkpoint request by messaging DBW*n* to begin writing dirty buffers. On completion of individual checkpoint requests, CKPT updates data file headers and control files to record most recent checkpoint.<br>CKPT checks every three seconds to see whether the amount of memory exceeds the value of the `PGA_AGGREGATE_LIMIT` initialization parameter, and if so, takes the action described in "PGA_AGGREGATE_LIMIT".<br>**See Also:** *Oracle Database Concepts* | Database instances, Oracle ASM instances |
| CP*nn* | Database Capture Process | Captures database changes from the redo log by using the infrastructure of LogMiner | The capture process name is CP*nn*, where *nn* can include letters and numbers. The underlying LogMiner process name is MS*nn*, where *nn* can include letters and numbers. The capture process includes one reader server that reads the redo log and divides it into regions, one or more preparer servers that scan the redo log, and one builder server that merges redo records from the preparer servers. Each reader server, preparer server, and builder server is a process. Query the `V$STREAMS_CAPTURE`, `V$XSTREAM_CAPTURE`, and `V$GOLDENGATE_CAPTURE` view for information about this background process.<br>**See Also:** *Oracle Streams Concepts and Administration* and *Oracle Database XStream Guide* | Database instances, XStream Outbound Servers, Oracle Streams |
| CS*nn* | I/O Calibration Process | Issues I/Os to storage as part of storage calibration. | CS*nn* slave processes are started on execution of the `DBMS_RESOURCE_MANAGER.CALIBRATE_IO()` procedure. There is one slave process per CPU on each node of the database. | Database instances, Oracle RAC |
| CTWR | Change Tracking Writer Process | Tracks changed data blocks as part of the Recovery Manager block change tracking feature | CTWR tracks changed blocks as redo is generated at a primary database and as redo is applied at a standby database. The process is slightly different depending on the type of database.<br>**See Also:** *Oracle Database Backup and Recovery User's Guide* | Database instances |
| CX*nn* | Streams Propagation Sender Process | Sends LCRs to a propagation receiver | The propagation sender process name is CX*nn*, where *nn* can include letters and numbers. In an Oracle Streams combined capture and apply optimization, the propagation sender sends LCRs directly to the propagation receiver to improve performance. The propagation receiver passes the LCRs to an apply process. Query `V` | Database instances, XStream Outbound Server, Oracle Streams |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | | `$PROPAGATION_SENDER` for information about a propagation sender. | |
| DBRM | Database Resource Manager Process | Sets resource plans and performs other tasks related to the Database Resource Manager | If a resource plan is not enabled, then this process is idle.<br>**See Also:** *Oracle Database Administrator's Guide* | Database instances |
| DBW*n* | Database Writer Process | Writes modified blocks from the database buffer cache to the data files | The primary responsibility of the Database Writer Process is to write data blocks to disk. It also handles checkpoints, file open synchronization, and logging of Block Written records.<br>In many cases the blocks that the Database Writer Process writes are scattered throughout the disk. Thus, the writes tend to be slower than the sequential writes performed by LGWR. The Database Writer Process performs multiblock writes when possible to improve efficiency. The number of blocks written in a multiblock write varies by operating system.<br>The `DB_WRITER_PROCESSES` initialization parameter specifies the number of Database Writer Processes. There can be 1 to 100 Database Writer Processes. The names of the first 36 Database Writer Processes are DBW0-DBW9 and DBWa-DBWz. The names of the 37th through 100th Database Writer Processes are BW36-BW99. The database selects an appropriate default setting for the `DB_WRITER_PROCESSES` parameter or adjusts a user-specified setting based on the number of CPUs and processor groups.<br>**See Also**: "DB_WRITER_PROCESSES" | Database instances |
| DIA0 | Diagnostic Process | Detects and resolves hangs and deadlocks | | Database instances, Oracle ASM instances |
| DIAG | Diagnostic Capture Process | Performs diagnostic dumps | DIAG performs diagnostic dumps requested by other processes and dumps triggered by process or instance termination. In Oracle RAC, DIAG performs global diagnostic dumps requested by remote instances. | Database instances, Oracle ASM instances |
| DM*nn* | Data Pump Master Process | Coordinates the Data Pump job tasks performed by Data Pump worker processes and handles client interactions | The Data Pump master (control) process is started during job creation and coordinates all tasks performed by the Data Pump job. It handles all client interactions and communication, establishes all job contexts, and coordinates all worker process activities on behalf of the job. | Database instances, Data Pump |
| DMON | Data Guard Broker | Manages and monitors a database that is part of a Data | When you start the Data Guard broker, a DMON process is created. DMON runs for every database instance that is managed by the broker. DMON | Database instances, |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | Monitor Process | Guard broker configuration | interacts with the local database and the DMON processes of the other databases to perform the requested function. DMON also monitors the health of the broker configuration and ensures that every database has a consistent description of the configuration.<br>DMON maintains profiles about all database objects in the broker configuration in a binary configuration file. A copy of this file is maintained by the DMON process for each of the databases that belong to the broker configuration. The process is created when the `DG_BROKER_START` initialization parameter is set to `true`.<br>**See Also:** *Oracle Data Guard Broker* | Data Guard |
| D*nnn* | Dispatcher Process | Performs network communication in the shared server architecture | In the shared server architecture, clients connect to a dispatcher process, which creates a virtual circuit for each connection. When the client sends data to the server, the dispatcher receives the data into the virtual circuit and places the active circuit on the common queue to be picked up by an idle shared server. The shared server then reads the data from the virtual circuit and performs the database work necessary to complete the request. When the shared server must send data to the client, the server writes the data back into the virtual circuit and the dispatcher sends the data to the client. After the shared server completes the client request, the server releases the virtual circuit back to the dispatcher and is free to handle other clients.<br>Several initialization parameters relate to shared servers. The principal parameters are: `DISPATCHERS`, `SHARED_SERVERS`, `MAX_SHARED_SERVERS`, `LOCAL_LISTENER`, `REMOTE_LISTENER`.<br>**See Also:** *Oracle Database Concepts* | Database instances, shared servers |
| DSKM | Slave Diskmon Process | Acts as the conduit between the database, Oracle ASM instances, and the Master Diskmon daemon to communicate information to Exadata storage | This process is active only if Exadata Storage is used. DSKM performs operations related to Exadata I/O fencing and Exadata cell failure handling. | Oracle ASM instances, Exadata |
| DW*nn* | Data Pump Worker Process | Performs Data Pump tasks as assigned by the Data Pump master process | The Data Pump worker process is responsible for performing tasks that are assigned by the Data Pump master process, such as the loading and unloading of metadata and data. | Database instances |
| EMNC | EMON Coordinator Process | Coordinates database event management and notifications | EMNC is a master background process that coordinates event management and notification activity in the database, including Streams Event | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | | Notifications, Continuous Query Notifications, and Fast Application Notifications. | |
| E*nnn* | EMON Slave Process | Performs database event management and notifications | The database event management and notification load is distributed among the EMON slave processes. These processes work on the system notifications in parallel, offering a capability to process a larger volume of notifications, a faster response time, and a lower shared memory use for staging notifications. | Database instances |
| FBDA | Flashback Data Archiver Process | Archives historical rows for tracked tables into flashback data archives and manages archive space, organization, and retention | When a transaction that modifies a tracked table commits, FBDA stores the pre-image of the rows in the archive. FBDA maintains metadata on the current rows and tracks how much data has been archived.<br>FBDA is also responsible for automatically managing the flashback data archive for space, organization (partitioning tablespaces), and retention. FBDA also keeps track of how far the archiving of tracked transactions has progressed.<br>**See Also:** *Oracle Database Development Guide* | Database instances |
| FD*nn* | Oracle ASM Stale FD Cleanup Slave Process | Cleans up Oracle ASM stale file descriptors on foreground processes | This process cleans up Oracle ASM stale file descriptors on foreground processes if an Oracle ASM disk is globally closed. | Database and Oracle ASM instances |
| FENC | Fence Monitor Process | Processes fence requests for RDBMS instances which are using Oracle ASM instances | CSS monitors RDBMS instances which are connected to the Oracle ASM instance and constantly doing I/Os. When the RDBMS instance terminates due to a failure, all the outstanding I/O's from the RDBMS instance should be drained and any new I/O's rejected. FENC receives and processes the fence request from CSSD. | Oracle ASM instances |
| FMON | File Mapping Monitor Process | Manages mapping information for the Oracle Database file mapping interface | The `DBMS_STORAGE_MAP` package enables you to control the mapping operations. When instructed by the user, FMON builds mapping information and stores it in the SGA, refreshes the information when a change occurs, saves the information to the data dictionary, and restores it to the SGA at instance startup.<br>FMON is started by the database whenever the `FILE_MAPPING` initialization parameter is set to `true`. | Database instances, Oracle ASM instances |
| FSFP | Data Guard Broker Fast Start Failover Pinger Process | Maintains fast-start failover state between the primary and target standby databases | FSFP is created when fast-start failover is enabled. | Database instances, Data Guard |
| GCR*n* | Global Conflict | Performs synchronous tasks on behalf of LMHB | GCR*n* processes are transient slaves that are started and stopped as required by LMHB to perform synchronous or resource intensive tasks. | Database instances, Oracle ASM |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | Resolution Slave Process | | | instances, Oracle RAC |
| GEN0 | General Task Execution Process | Performs required tasks including SQL and DML | | Database instances, Oracle ASM instances, Oracle ASM Proxy instances |
| GMON | ASM Disk Group Monitor Process | Monitors all mounted Oracle ASM disk groups | GMON monitors all the disk groups mounted in an Oracle ASM instance and is responsible for maintaining consistent disk membership and status information. Membership changes result from adding and dropping disks, whereas disk status changes result from taking disks offline or bringing them online. | Oracle ASM instances |
| GTX*n* | Global Transaction Process | Provides transparent support for XA global transactions in an Oracle RAC environment | These processes help maintain the global information about XA global transactions throughout the cluster. Also, the processes help perform two-phase commit for global transactions anywhere in the cluster so that an Oracle RAC database behaves as a single system to the externally coordinated distributed transactions.<br>The `GLOBAL_TXN_PROCESSES` initialization parameter specifies the number of GTX*n* processes, where *n* is 0-9 or a-j. The database automatically tunes the number of these processes based on the workload of XA global transactions. You can disable these processes by setting the parameter to 0. If you try to run XA global transactions with these processes disabled, an error is returned.<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle RAC |
| I*nnn* | Disk and Tape I/O Slave Process | Serves as an I/O slave process spawned on behalf of DBWR, LGWR, or an RMAN backup session | I/O slave process can be configured on platforms where asynchronous I/O support is not available. These slaves are started by setting the corresponding slave enable parameter in the server parameter file. The I/O slaves simulate the asynchronous I/O behavior when the underlying platform does not have native support for asynchronous I/O. | Database instances |
| IMCO | In-memory Database Process | Initiates background population and repopulation of in-memory enabled objects | The IMCO background process initiates population (prepopulation) of in-memory enabled objects with priority LOW/MEDIUM/HIGH/CRITICAL. In-memory enabled objects with priority NONE will not be prepopulated but will be populated on demand via W*nnn* processes | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | | when queried. The IMCO background process can also initiate repopulation of in-memory objects. | |
| INSV | Data Guard Broker Instance Slave Process | Performs Data Guard broker communication among instances in an Oracle RAC environment | INSV is created when the `DG_BROKER_START` initialization parameter is set to `true`. | Database instances, Data Guard |
| IPC0 | IPC Service Background Process | Common background server for basic messaging and RDMA primitives based on IPC (Inter-process communication) methods. | IPC0 handles very high rates of incoming connect requests, as well as, completing reconfigurations to support basic messaging and RDMA primitives over several transports such as UDP, RDS, InfiniBand and RC. | Oracle RAC |
| J*nnn* | Job Queue Slave Process | Executes jobs assigned by the job coordinator | Job slave processes are created or awakened by the job coordinator when it is time for a job to be executed.<br>Job slaves gather all the metadata required to run the job from the data dictionary. The slave processes start a database session as the owner of the job, execute triggers, and then execute the job. After the job is complete, the slave processes commit and then execute appropriate triggers and close the session. The slave can repeat this operation in case additional jobs need to be run. | Database instances |
| LCK*n* | Lock Process | Manages global enqueue requests and cross-instance broadcasts | The process handles all requests for resources other than data blocks. For examples, LCK*n* manages library and row cache requests. Possible processes are LCK0 and LCK1. | Database instances, Oracle ASM instances, Oracle RAC |
| LDD*n* | Global Enqueue Service Daemon Helper Slave | Helps the LMD*n* processes with various tasks | LDD*n* processes are slave processes spawned on demand by LMD*n* processes. They are spawned to help the dedicated LMD*n* processes with various tasks when certain workloads start creating performance bottlenecks. These slave processes are transient as they are started on demand and they can be shutdown when no longer needed. There can be up to 36 of these slave processes (LDD0-LDDz). | Database instances, Oracle ASM instances, Oracle RAC |
| LG*nn* | Log Writer Worker | Writes redo log | On multiprocessor systems, LGWR creates worker processes to improve the performance of writing to the redo log. LGWR workers are not used when there is a SYNC standby destination. Possible processes include LG00-LG99. | Database instances |
| LGWR | Log Writer Process | Writes redo entries to the online redo log | Redo log entries are generated in the redo log buffer of the system global area (SGA). LGWR writes the redo log entries sequentially into a redo | Database instances, Oracle |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | | log file. If the database has a multiplexed redo log, then LGWR writes the redo log entries to a group of redo log files.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Administrator's Guide* | ASM instances |
| LMD*n* | Global Enqueue Service Daemon Process | Manages incoming remote resource requests from other instances | LMD*n* processes enqueue resources managed under Global Enqueue Service. In particular, they process incoming enqueue request messages and control access to global enqueues. They also perform distributed deadlock detections. There can be up to 36 of these processes (LMD0-LMDz). | Database instances, Oracle ASM instances, Oracle RAC |
| LMHB | Global Cache/ Enqueue Service Heartbeat Monitor | Monitor the heartbeat of several processes | LMHB monitors the CKPT, DIA*n*, LCK*n*, LG*nn*, LGWR, LMD*n*, LMON, LMS*n* , and RMS*n* processes to ensure they are running normally without blocking or spinning. | Database instances, Oracle ASM instances, Oracle RAC |
| LMON | Global Enqueue Service Monitor Process | Monitors an Oracle RAC cluster to manage global resources | LMON maintains instance membership within Oracle RAC. The process detects instance transitions and performs reconfiguration of GES and GCS resources.<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle ASM instances, Oracle RAC |
| LMS*n* | Global Cache Service Process | Manages resources and provides resource control among Oracle RAC instances | LMS, where *n* is 0-9 or a-z, maintains a lock database for Global Cache Service (GCS) and buffer cache resources. This process receives, processes, and sends GCS requests, block transfers, and other GCS-related messages.<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle ASM instances, Oracle RAC |
| LREG | Listener Registration Process | Registers the instance with the listeners | LREG notifies the listeners about instances, services, handlers, and endpoint. | Database instances, Oracle ASM instances, Oracle RAC |
| LSP0 | Logical Standby Coordinator Process | Schedules transactions for Data Guard SQL Apply | LSP0 is the initial process created upon startup of Data Guard SQL Apply. In addition to managing LogMiner and Apply processes, LSP0 is responsible for maintaining inter-transaction dependencies and appropriately scheduling transactions with applier processes. LSP0 is also responsible for detecting and enabling run-time parameter changes for the SQL Apply product as a whole. | Database instances, Data Guard |
| LSP1 | Logical Standby | Performs a logical standby dictionary | The LSP1 process is spawned on a logical standby database that is intended to become the new | Database instances, |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | Dictionary Build Process | build on a primary database | primary database. A logical standby database becomes a primary database because of switchover or failover. The dictionary is necessary for logical standby databases to interpret the redo of the new primary database. | Data Guard |
| LSP2 | Logical Standby Set Guard Process | Determines which database objects will be protected by the database guard | The LSP2 process is created as needed during startup of SQL Apply to update the list of objects that are protected by the database guard. | Database instances, Data Guard |
| L*nnn* | Pooled Server Process | Handles client requests in Database Resident Connection Pooling | In Database Resident Connection Pooling, clients connect to a connection broker process. When a connection becomes active, the connection broker hands off the connection to a compatible pooled server process. The pooled server process performs network communication directly on the client connection and processes requests until the client releases the server. After being released, the connection is returned to the broker for monitoring, leaving the server free to handle other clients.<br>**See Also:** *Oracle Database Concepts* | Database instances, Database Resident Connection Pooling |
| MARK | Mark AU for Resynchronization Coordinator Process | Marks ASM allocation units as stale following a missed write to an offline disk | MARK essentially tracks which extents require resynchronization for offline disks. This process runs in the database instance and is started when the database instance first begins using the Oracle ASM instance. If required, MARK can also be started on demand when disks go offline in the Oracle ASM redundancy disk group. | Database instances, Oracle ASM instances |
| MMAN | Memory Manager Process | Serves as the instance memory manager | This process performs the resizing of memory components on the instance. | Database instances, Oracle ASM instances |
| MMNL | Manageability Monitor Lite Process | Performs tasks relating to manageability, including active session history sampling and metrics computation | MMNL performs many tasks relating to manageability, including session history capture and metrics computation. | Database instances, Oracle ASM instances |
| MMON | Manageability Monitor Process | Performs or schedules many manageability tasks | MMON performs many tasks related to manageability, including taking Automatic Workload Repository snapshots and performing Automatic Database Diagnostic Monitor analysis. | Database instances, Oracle ASM instances |
| M*nnn* | MMON Slave Process | Performs manageability tasks on behalf of MMON | M*nnn* performs manageability tasks dispatched to them by MMON. Tasks performed include taking Automatic Workload Repository snapshots and Automatic Database Diagnostic Monitor analysis. | Database instances, Oracle ASM instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| MRP0 | Managed Standby Recovery Process | Coordinates the application of redo on a physical standby database | MRP0 is spawned at the start of redo apply on a physical standby database. This process handles the extraction of redo and coordinates the application of that redo on a physical standby database.<br>**See Also:** *Oracle Data Guard Concepts and Administration* | Database instances, Data Guard |
| MS*nn* | LogMiner Worker Process | Reads redo log files and translates and assembles into transactions | Multiple MS*nn* processes can exists, where *n* is 0-9 or a-Z. A minimum of three MS*nn* processes work as a group to provide transactions to a LogMiner client, for example, a logical standby database or a database capture. There may be more than one such group, for example, multiple capture processes configured for either local or downstream capture in a database. | Database instances, Logical Standby, Oracle Streams, XStream Outbound servers, Oracle GoldenGate |
| N*nnn* | Connection Broker Process | Monitors idle connections and hands off active connections in Database Resident Connection Pooling | In Database Resident Connection Pooling, clients connect to a connection broker process. When a connection becomes active, the connection broker hands off the connection to a compatible pooled server process. The pooled server process performs network communication directly on the client connection and processes requests until the client releases the server. After being released, the connection is returned to the broker for monitoring, leaving the server free to handle other clients.<br>**See Also:** *Oracle Database Concepts* | Database instances, Database Resident Connection Pooling |
| NSS*n* | Network Server SYNC Process | Transfers redo from current online redo logs to remote standby destinations configured for SYNC transport | NSS*n* can run as multiple processes, where *n* is 1-9 or A.<br>**See Also:** *Oracle Data Guard Concepts and Administration* | Database instances, Data Guard |
| NSV*n* | Data Guard Broker NetSlave Process | Performs broker network communications between databases in a Data Guard environment | NSV*n* is created when a Data Guard broker configuration is enabled. There can be as many NSV*n* processes (where *n* is 0- 9 and A-U) created as there are databases in the Data Guard broker configuration. | Database instances, Data Guard |
| OCF*n* | ASM CF Connection Pool Process | Maintains a connection to the Oracle ASM instance for metadata operations | | Database instances, Oracle ASM instances |
| OFSD | Oracle File Server | Serves file system requests submitted to an Oracle instance | This background process listens for new file system requests, both management (like mount, | Database instances, |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | Background Process | | unmount, and export) and I/O requests, and executes them using Oracle threads. | Oracle RAC |
| O*nnn* | ASM Connection Pool Process | Maintains a connection to the Oracle ASM instance for metadata operations | O*nnn* slave processes are spawned on demand. These processes communicate with the Oracle ASM instance. | Database instances, Oracle ASM instances |
| PING | Interconnect Latency Measurement Process | Assesses latencies associated with communications for each pair of cluster instances | Every few seconds, the process in one instance sends messages to each instance. The message is received by PING on the target instance. The time for the round trip is measured and collected. | Database instances, Oracle ASM instances, Oracle RAC |
| PMON | Process Monitor | Monitors the other background processes and performs process recovery when a server or dispatcher process terminates abnormally | PMON periodically performs cleanup of all the following:<br>• Processes that died abnormally<br>• Sessions that were killed<br>• Detached transactions that have exceeded their idle timeout<br>• Detached network connections which have exceeded their idle timeout<br>In addition, PMON monitors, spawns, and stops the following as needed:<br>• Dispatcher and shared server processes<br>• Job queue processes<br>• Pooled server processes for database resident connection pooling<br>• Restartable background processes<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Net Services Administrator's Guide* | Database instances, Oracle ASM instances, Oracle ASM Proxy instances |
| P*nnn* | Parallel Query Slave Process | Perform parallel execution of a SQL statement (query, DML, or DDL) | Parallel Query has two components: a foreground process that acts as query coordinator and a set of parallel slaves (P*nnn*) that are background processes. These background processes are spawned or reused during the start of a parallel statement. They receive and perform units of work sent from the query coordinator.<br>The maximum number of P*nnn* processes is controlled by the initialization parameter `PARALLEL_MAX_SERVERS`. Slave processes are numbered from 0 to the `PARALLEL_MAX_SERVERS` setting. If the query is a `GV$` query, then these background processes are numbered backward, starting from PPA7. | Database instances, Oracle ASM instances |
| PR*nn* | Parallel Recovery Process | Performs tasks assigned by the coordinator process performing parallel recovery | PR*nn* serves as a slave process for the coordinator process performing parallel media recovery and carries out tasks assigned by the coordinator. The default number of these processes is based on number of CPUs. | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| PSP0 | Process Spawner Process | Spawns Oracle background processes after initial instance startup | | Database instances, Oracle ASM instances |
| QMNC | Non-sharded queue master process | Monitors AQ | QMNC is the non-sharded queue master process responsible for facilitating various background activities required by AQ and Oracle Streams: time management of messages, management of nonpersistent queues, cleanup of resources, and so on. QMNC dynamically spawns Q*nnn* processes as needed for performing these tasks.<br>Note that if the `AQ_TM_PROCESSES` initialization parameter is set to 0, this process will not start. The database writes the following message to the alert log: `WARNING: AQ_TM_PROCESSES is set to 0. System might be adversely affected`. | Database instances Advanced Queueing |
| QM*nn* | AQ Master Class Process | Per instance per AQ Master Class Process | Each of this type of process represents a single class of work item such as AQ notification, queue monitors, and cross process. | Database instances Advanced Queueing |
| Q*nnn* | AQ Server Class Process | Per AQ Master Class server process | Each server class process acts on behalf of an AQ master class process. This relationship is maintained until the master requires services of a particular service process. Once released, the server class processes are moved to a free server pool. | Database instances Advanced Queueing |
| RBAL | ASM Rebalance Master Process | Coordinates rebalance activity | In an Oracle ASM instance, it coordinates rebalance activity for disk groups. In a database instance, it manages Oracle ASM disk groups. | Database instances, Oracle ASM instances |
| RCBG | Result Cache Background Process | Handles result cache messages | This process is used for handling invalidation and other messages generated by server processes attached to other instances in Oracle RAC. | Database instances, Oracle RAC |
| RECO | Recoverer Process | Resolves distributed transactions that are pending because of a network or system failure in a distributed database | RECO uses the information in the pending transaction table to finalize the status of in-doubt transactions. At timed intervals, the local RECO attempts to connect to remote databases and automatically complete the commit or rollback of the local portion of any pending distributed transactions. All transactions automatically resolved by RECO are removed from the pending transaction table.<br>**See Also:** *Oracle Database Concepts* and *Oracle Database Net Services Administrator's Guide* | Database instances |
| RM | RAT Masking Slave Process | Extracts and masks bind values from workloads like SQL | This background process is used with Data Masking and Real Application Testing. | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | tuning sets and DB Replay capture files | | |
| RMON | Rolling Migration Monitor Process | Manages the rolling migration procedure for an Oracle ASM cluster | The RMON process is spawned on demand to run the protocol for transitioning an ASM cluster in and out of rolling migration mode. | Oracle ASM instance, Oracle RAC |
| RMS*n* | Oracle RAC Management Process | Performs manageability tasks for Oracle RAC | RMS*n* performs a variety of tasks, including creating resources related to Oracle RAC when new instances are added to a cluster.<br>**See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* | Database instances, Oracle RAC |
| RMV*n* | Global Cache Service Remaster Process | Performs remastering for cluster reconfiguration and dynamic remastering | Each RMV is a slave process for LMS*n* to handle remastering work. They are also helper processes for LMS to handle non-critical work from global cache service. | Database instances, Oracle RAC |
| R*nnn* | ASM Block Remap Slave Process | Remaps a block with a read error | A database instance reading from an Oracle ASM disk group can encounter an error during a read. If possible, Oracle ASM asynchronously schedules a R*nnn* slave process to remap this bad block from a mirror copy. | Oracle ASM instances |
| RP*nn* | Capture Processing Worker Process | Processes a set of workload capture files | RP*nn* are worker processes spawned by calling `DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE(capture_dir,parallel_level)`. Each worker process is assigned a set of workload capture files to process.<br>Worker processes execute in parallel without needing to communicate with each other. After each process is finished processing its assigned files, it exits and informs its parent process.<br>The number of worker processes is controlled by the `parallel_level` parameter of `DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE`. By default, `parallel_level` is null. Then, the number of worker processes is computed as follows:<br>`SELECT VALUE FROM V$PARAMETER WHERE NAME='cpu_count';`<br>When `parallel_level` is `1`, no worker processes are spawned. | Database instances |
| RPOP | Instant Recovery Repopulation Daemon | Responsible for re-creating and/or repopulating data files from snapshot files and backup files | The RPOP process is responsible for re-creating and repopulating data files from snapshots files. It works with the instant recovery feature to ensure immediate data file access. The local instance has immediate access to the remote snapshot file's data, while repopulation of the recovered primary data files happens concurrently. Any changes in the data are managed between the instance's DBW | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | | processes and RPOP to ensure the latest copy of the data is returned to the user. | |
| RSM0 | Data Guard Broker Worker Process | Performs monitoring management tasks related to Data Guard on behalf of DMON | The process is created when a Data Guard broker configuration is enabled. | Database instances, Data Guard |
| RSMN | Remote Slave Monitor Process | Manages background slave process creation and communication on remote instances in Oracle RAC | This background process manages the creation of slave processes and the communication with their coordinators and peers. These background slave processes perform tasks on behalf of a coordinating process running in another cluster instance. | Database instances, Oracle RAC |
| RVWR | Recovery Writer Process | Writes flashback data to the flashback logs in the fast recovery area | RVWR writes flashback data from the flashback buffer in the SGA to the flashback logs. RVWR also creates flashback logs and performs some tasks for flashback log automatic management. | Database instances, Flashback Database |
| SA*nn* | SGA Allocator | Allocates SGA | A small fraction of SGA is allocated during instance startup. The SA*nn* process allocates the rest of SGA in small chunks. The process exits upon completion of SGA allocation.<br>The possible processes are SA00 - SAzz. | Database instances |
| SCC*n* | ASM Disk Scrubbing Slave Check Process | Performs Oracle ASM disk scrubbing check operation | SCC*n* acts as a slave process for SCRB and performs the checking operations. The possible processes are SCC0-SCC9. | Oracle ASM instances |
| SCRB | ASM Disk Scrubbing Master Process | Coordinates Oracle ASM disk scrubbing operations | SCRB runs in an Oracle ASM instance and coordinates Oracle ASM disk scrubbing operations. | Oracle ASM instances |
| SCR*n* | ASM Disk Scrubbing Slave Repair Process | Performs Oracle ASM disk scrubbing repair operation | SCR*n* acts as a slave process for SCRB and performs the repairing operations. The possible processes are SCR0-SCR9. | Oracle ASM instances |
| SCV*n* | ASM Disk Scrubbing Slave Verify Process | Performs Oracle ASM disk scrubbing verify operation | SCV*n* acts as a slave process for SCRB and performs the verifying operations. The possible processes are SCV0-SCV9. | Oracle ASM instances |
| SMCO | Space Management Coordinator Process | Coordinates the execution of various space management tasks | This background process coordinates the execution of various space management tasks, including proactive space allocation and space reclamation. SMCO dynamically spawns slave processes (W*nnn*) to implement these tasks. | Database instances |
| SMON | System Monitor Process | Performs critical tasks such as instance recovery and dead transaction recovery, and maintenance tasks such as | SMON performs many database maintenance tasks, including the following:<br>• Creates and manages the temporary tablespace metadata<br>• Reclaims space used by orphaned temporary segments | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | temporary space reclamation, data dictionary cleanup, and undo tablespace management | • Maintains the undo tablespace by onlining, offlining, and shrinking the undo segments based on undo space usage statistics<br>• Cleans up the data dictionary when it is in a transient and inconsistent state<br>• Maintains the SCN to time mapping table used to support Oracle Flashback features<br>In an Oracle RAC database, the SMON process of one instance can perform instance recovery for other instances that have failed.<br>SMON is resilient to internal and external errors raised during background activities.<br>**See Also:** *Oracle Database Concepts* | |
| S*nnn* | Shared Server Process | Handles client requests in the shared server architecture | In the shared server architecture, clients connect to a dispatcher process, which creates a virtual circuit for each connection. When the client sends data to the server, the dispatcher receives the data into the virtual circuit and places the active circuit on the common queue to be picked up by an idle shared server. The shared server then reads the data from the virtual circuit and performs the database work necessary to complete the request. When the shared server must send data to the client, the server writes the data back into the virtual circuit and the dispatcher sends the data to the client. After the shared server completes the client request, the server releases the virtual circuit back to the dispatcher and is free to handle other clients.<br>Several initialization parameters relate to shared servers. The principal parameters are: `DISPATCHERS`, `SHARED_SERVERS`, `MAX_SHARED_SERVERS`, `LOCAL_LISTENER`, `REMOTE_LISTENER`.<br>**See Also:** *Oracle Database Concepts* | Database instances, shared servers |
| TEM*n* | ASM disk Test Error Emulation Process | Emulates I/O errors on Oracle ASM disks through named events | I/O errors can be emulated on Oracle ASM disk I/O through named events. The scope can be the process, instance, or even cluster. Optionally, a set of AUs can be chosen for error emulation. | Oracle ASM instances |
| TT*nn* | Redo Transport Slave Process | Ships redo from current online and standby redo logs to remote standby destinations configured for ASYNC transport | TT*nn* can run as multiple processes, where *nn* is 00 to ZZ.<br>**See Also:** *Oracle Data Guard Concepts and Administration* | Database instances, Data Guard |
| U*nnn* | Container process for threads | Host processes where database processes execute as threads. | U*nnn* processes are database container operating system processes where database backgrounds processes like SMON, CJQ0, and database foreground processes run. The `V$PROCESS` view lists database processes running in these container processes. These container processes are created only when the `THREADED_EXECUTION` | Database instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| | | | initialization parameter is set to TRUE. The number of these processes vary depending on the active database processes. On a host with multiple NUMA nodes, there will be at least one U*nnn* process per NUMA node.<br>These processes are fatal processes, if any of them is killed, it will result in instance termination. These processes exit when the instance is shut down or terminated. | |
| VBG*n* | Volume Background Process | Communicates between the Oracle ASM instance and the operating system volume driver | VBG*n* handles messages originating from the volume driver in the operating system and sends them to the Oracle ASM instance.<br>VBG*n* can run as multiple processes, where *n* is 0-9. | Oracle ASM instances, Oracle ASM Proxy instances |
| VDBG | Volume Driver Process | Forwards Oracle ASM requests to perform various volume-related tasks | VDBG handles requests to lock or unlock an extent for rebalancing, volume resize, disk offline, add or drop a disk, force and dismount disk group to the Dynamic Volume Manager driver. | Oracle ASM instances, Oracle ASM Proxy instances |
| VIO0, VIO1, VIO2, VIO3 | Volume I/O | Route ADVM volume I/O for ASM instances on compute nodes within an Exadata | These processes handle requests for I/Os targeted at storage not locally accessible. They are used for Exadata targeted storage as well. These background processes only start when an ASM Volume is created and set up to be used. One process will start for each NUMA node on target machines. Under normal operation on non-Exadata hardware and on Exadata hardware that is not utilizing ASM volumes, these processes will not be started. | Oracle ASM Proxy instances |
| VKRM | Virtual Scheduler for Resource Manager Process | Serves as centralized scheduler for Resource Manager activity | VKRM manages the CPU scheduling for all managed Oracle processes. The process schedules managed processes in accordance with an active resource plan. | Database instances |
| VKTM | Virtual Keeper of Time Process | Provides a wall clock time and reference time for time interval measurements | VKTM acts as a time publisher for an Oracle instance. VKTM publishes two sets of time: a wall clock time using a seconds interval and a higher resolution time (which is not wall clock time) for interval measurements. The VKTM timer service centralizes time tracking and offloads multiple timer calls from other clients. | Database instances, Oracle ASM instances |
| VMB0 | Volume Membership Process | Maintains cluster membership on behalf of the Oracle ASM volume driver | This process membership in the cluster as an I/O-capable client on behalf of the Oracle ASM volume driver. | Oracle ASM instances, Oracle ASM Proxy instances |

| Name | Expanded Name | Short Description | Long Description | External Properties |
|---|---|---|---|---|
| VUBG | Volume drive Umbilicus Background | Relays messages between Oracle ASM instance and Oracle ASM Proxy instance that is used by ADVM (for ACFS) | | Oracle ASM instances, Oracle ASM Proxy instances |
| W*nnn* | Space Management Slave Process | Performs various background space management tasks, including proactive space allocation and space reclamation | W*nnn* slave processes perform work on behalf of Space Management and on behalf of the Oracle In-Memory Option.<br>• W*nnn* processes are slave processes dynamically spawned by SMCO to perform space management tasks in the background. These tasks include preallocating space into locally managed tablespace and SecureFiles segments based on space usage growth analysis, and reclaiming space from dropped segments. After being started, the slave acts as an autonomous agent. After it finishes task execution, it automatically picks up another task from the queue. The process terminates itself after being idle for a long time.<br>• W*nnn* processes execute in-memory populate and in-memory repopulate tasks for population or repopulation of in-memory enabled objects. For in-memory, both the IMCO background process and foreground processes will utilize W*nnn* slaves for population and repopulation. W*nnn* processes are utilized by the IMCO background process for prepopulation of in-memory enabled objects with priority LOW/MEDIUM/HIGH/CRITICAL, and for repopulation of in-memory objects. In-memory populate and repopulate tasks running on W*nnn* slaves are also initiated from foreground processes in response to queries and DMLs that reference in-memory enabled objects. | Database instances |
| XDMG | Exadata Automation Manager | Initiates automation tasks involved in managing Exadata storage | XDMG monitors all configured Exadata cells for state changes, such as a bad disk getting replaced, and performs the required tasks for such events. Its primary tasks are to watch for when inaccessible disks and cells become accessible again, and to initiate the ASM ONLINE operation. The ONLINE operation is handled by XDWK. | Oracle ASM instances, Exadata |
| XDWK | Exadata Automation Manager | Performs automation tasks requested by XDMG | XDWK gets started when asynchronous actions such as ONLINE, DROP, and ADD an Oracle ASM disk are requested by XDMG. After a 5 minute period of inactivity, this process will shut itself down. | Oracle ASM instances, Exadata |
| X*nnn* | ASM Disk Expel Slave Process | Performs Oracle ASM post-rebalance activities | This process expels dropped disks after an Oracle ASM rebalance. | Oracle ASM instances |

# Index

## A


ACTIVE_INSTANCE_COUNT initialization parameter, *1-23*
ALL_ALL_TABLES view, *2-7*
ALL_APPLY view, *2-12*
ALL_APPLY_CHANGE_HANDLERS view, *2-14*
ALL_APPLY_CONFLICT_COLUMNS view, *2-15*
ALL_APPLY_DML_HANDLERS view, *2-16*
ALL_APPLY_ENQUEUE view, *2-17*
ALL_APPLY_ERROR view, *2-17*
ALL_APPLY_ERROR_MESSAGES view, *2-19*
ALL_APPLY_EXECUTE view, *2-21*
ALL_APPLY_INSTANTIATED_GLOBAL view, *2-21*
ALL_APPLY_INSTANTIATED_OBJECTS view, *2-22*
ALL_APPLY_INSTANTIATED_SCHEMAS view, *2-22*
ALL_APPLY_KEY_COLUMNS view, *2-23*
ALL_APPLY_PARAMETERS view, *2-23*
ALL_APPLY_PROGRESS view, *2-24*
ALL_APPLY_TABLE_COLUMNS view, *2-25*
ALL_ARGUMENTS view, *2-26*
ALL_ASSEMBLIES view, *2-28*
ALL_ASSOCIATIONS view, *2-29*
ALL_ATTRIBUTE_TRANSFORMATIONS view, *2-30*
ALL_AUDIT_POLICIES view, *2-31*
ALL_AUDIT_POLICY_COLUMNS view, *2-32*
ALL_AW_PS view, *2-33*
ALL_AWS view, *2-34*
ALL_BASE_TABLE_MVIEWS view, *2-35*
ALL_CAPTURE view, *2-35*
ALL_CAPTURE_EXTRA_ATTRIBUTES view, *2-38*
ALL_CAPTURE_PARAMETERS view, *2-39*
ALL_CAPTURE_PREPARED_DATABASE view, *2-40*
ALL_CAPTURE_PREPARED_SCHEMAS view, *2-40*
ALL_CAPTURE_PREPARED_TABLES view, *2-41*
ALL_CATALOG view, *2-42*
ALL_CLUSTER_HASH_EXPRESSIONS view, *2-43*
ALL_CLUSTERING_DIMENSIONS view, *2-43*
ALL_CLUSTERING_JOINS view, *2-44*
ALL_CLUSTERING_KEYS view, *2-45*
ALL_CLUSTERING_TABLES view, *2-46*
ALL_CLUSTERS view, *2-48*
ALL_CODE_ROLE_PRIVS view, *2-50*
ALL_COL_COMMENTS view, *2-50*
ALL_COL_PENDING_STATS view, *2-51*
ALL_COL_PRIVS view, *2-52*
ALL_COL_PRIVS_MADE view, *2-53*
ALL_COL_PRIVS_RECD view, *2-54*
ALL_COLL_TYPES view, *2-54*
ALL_CONS_COLUMNS view, *2-55*
ALL_CONS_OBJ_COLUMNS view, *2-56*
ALL_CONSTRAINTS view, *2-57*
ALL_CONTEXT view, *2-59*
ALL_CREDENTIALS view, *2-60*
ALL_CUBE_ATTR_VISIBILITY view, *2-61*
ALL_CUBE_ATTRIBUTES view, *2-62*
ALL_CUBE_BUILD_PROCESSES view, *2-63*
ALL_CUBE_CALCULATED_MEMBERS view, *2-64*
ALL_CUBE_DIM_LEVELS view, *2-65*
ALL_CUBE_DIM_MODELS view, *2-65*
ALL_CUBE_DIM_VIEW_COLUMNS view, *2-66*
ALL_CUBE_DIM_VIEWS view, *2-67*
ALL_CUBE_DIMENSIONALITY view, *2-67*
ALL_CUBE_DIMENSIONS view, *2-68*
ALL_CUBE_HIER_LEVELS view, *2-69*
ALL_CUBE_HIER_VIEW_COLUMNS view, *2-70*
ALL_CUBE_HIER_VIEWS view, *2-71*
ALL_CUBE_HIERARCHIES view, *2-72*
ALL_CUBE_MEASURES view, *2-73*
ALL_CUBE_NAMED_BUILD_SPECS view, *2-74*
ALL_CUBE_SUB_PARTITION_LEVELS view, *2-75*
ALL_CUBE_VIEW_COLUMNS view, *2-75*
ALL_CUBE_VIEWS view, *2-76*
ALL_CUBES view, *2-77*
ALL_DB_LINKS view, *2-78*
ALL_DEF_AUDIT_OPTS view, *2-78*
ALL_DEPENDENCIES view, *2-80*
ALL_DEQUEUE_QUEUES view, *2-81*
ALL_DIM_ATTRIBUTES view, *2-81*
ALL_DIM_CHILD_OF view, *2-82*
ALL_DIM_HIERARCHIES view, *2-83*
ALL_DIM_JOIN_KEY view, *2-83*
ALL_DIM_LEVEL_KEY view, *2-84*
ALL_DIM_LEVELS view, *2-84*

ALL_DIMENSIONS view, 2-85
ALL_DIRECTORIES view, 2-86
ALL_EDITION_COMMENTS view, 2-86
ALL_EDITIONING_VIEW_COLS view, 2-87
ALL_EDITIONING_VIEW_COLS_AE view, 2-87
ALL_EDITIONING_VIEWS view, 2-88
ALL_EDITIONING_VIEWS_AE view, 2-89
ALL_EDITIONS view, 2-89
ALL_ENCRYPTED_COLUMNS view, 2-90
ALL_ERROR_TRANSLATIONS view, 2-90
ALL_ERRORS view, 2-91
ALL_ERRORS_AE view, 2-92
ALL_EVALUATION_CONTEXT_TABLES view, 2-93
ALL_EVALUATION_CONTEXT_VARS view, 2-94
ALL_EVALUATION_CONTEXTS view, 2-94
ALL_EXTERNAL_LOCATIONS view, 2-95
ALL_EXTERNAL_TABLES view, 2-96
ALL_FILE_GROUP_EXPORT_INFO view, 2-97
ALL_FILE_GROUP_FILES view, 2-97
ALL_FILE_GROUP_TABLES view, 2-98
ALL_FILE_GROUP_TABLESPACES view, 2-99
ALL_FILE_GROUP_VERSIONS view, 2-100
ALL_FILE_GROUPS view, 2-100
ALL_GG_INBOUND_PROGRESS view, 2-101
ALL_GOLDENGATE_INBOUND view, 2-102
ALL_GOLDENGATE_PRIVILEGES view, 2-103
ALL_GOLDENGATE_RULES view, 2-104
ALL_HEAT_MAP_SEG_HISTOGRAM view, 2-105
ALL_HEAT_MAP_SEGMENT view, 2-106
ALL_HISTOGRAMS synonym for ALL_TAB_HISTOGRAMS, 2-107
ALL_IDENTIFIERS view, 2-107
ALL_IND_COLUMNS view, 2-108
ALL_IND_EXPRESSIONS view, 2-109
ALL_IND_PARTITIONS view, 2-110
ALL_IND_PENDING_STATS view, 2-113
ALL_IND_STATISTICS view, 2-114
ALL_IND_SUBPARTITIONS view, 2-116
ALL_INDEXES view, 2-119
ALL_INDEXTYPE_ARRAYTYPES view, 2-124
ALL_INDEXTYPE_COMMENTS view, 2-125
ALL_INDEXTYPE_OPERATORS view, 2-125
ALL_INDEXTYPES view, 2-126
ALL_INTERNAL_TRIGGERS view, 2-127
ALL_JAVA_ARGUMENTS view, 2-127
ALL_JAVA_CLASSES view, 2-128
ALL_JAVA_COMPILER_OPTIONS view, 2-129
ALL_JAVA_DERIVATIONS view, 2-130
ALL_JAVA_FIELDS view, 2-130
ALL_JAVA_IMPLEMENTS view, 2-132
ALL_JAVA_INNERS view, 2-132
ALL_JAVA_LAYOUTS view, 2-134
ALL_JAVA_METHODS view, 2-134
ALL_JAVA_NCOMPS view, 2-136
ALL_JAVA_RESOLVERS view, 2-137
ALL_JAVA_THROWS view, 2-137
ALL_JOBS synonym for USER_JOBS view, 2-138
ALL_JOIN_IND_COLUMNS view, 2-138
ALL_JSON_COLUMNS view, 2-139
ALL_LIBRARIES view, 2-140
ALL_LOB_PARTITIONS view, 2-141
ALL_LOB_SUBPARTITIONS view, 2-143
ALL_LOB_TEMPLATES view, 2-146
ALL_LOBS view, 2-147
ALL_LOG_GROUP_COLUMNS view, 2-149
ALL_LOG_GROUPS view, 2-150
ALL_MEASURE_FOLDER_CONTENTS view, 2-151
ALL_MEASURE_FOLDER_SUBFOLDERS view, 2-151
ALL_MEASURE_FOLDERS view, 2-152
ALL_METADATA_PROPERTIES view, 2-153
ALL_METHOD_PARAMS view, 2-153
ALL_METHOD_RESULTS view, 2-154
ALL_MINING_MODEL_ATTRIBUTES view, 2-155
ALL_MINING_MODEL_SETTINGS view, 2-157
ALL_MINING_MODELS view, 2-157
ALL_MP_GRAPH_WORKSPACES view, 2-5
ALL_MP_PARENT_WORKSPACES view, 2-5
ALL_MVIEW_AGGREGATES view, 2-159
ALL_MVIEW_ANALYSIS view, 2-160
ALL_MVIEW_COMMENTS view, 2-161
ALL_MVIEW_DETAIL_PARTITION view, 2-162
ALL_MVIEW_DETAIL_RELATIONS view, 2-163
ALL_MVIEW_DETAIL_SUBPARTITION view, 2-164
ALL_MVIEW_JOINS view, 2-165
ALL_MVIEW_KEYS view, 2-166
ALL_MVIEW_LOGS view, 2-166
ALL_MVIEW_REFRESH_TIMES view, 2-168
ALL_MVIEWS view, 2-168
ALL_NESTED_TABLE_COLS view, 2-173
ALL_NESTED_TABLES view, 2-175
ALL_OBJ_COLATTRS view, 2-176
ALL_OBJECT_TABLES view, 2-177
ALL_OBJECTS view, 2-181
ALL_OBJECTS_AE view, 2-183
ALL_OPANCILLARY view, 2-184
ALL_OPARGUMENTS view, 2-185
ALL_OPBINDINGS view, 2-185
ALL_OPERATOR_COMMENTS view, 2-187
ALL_OPERATORS view, 2-187
ALL_OUTLINE_HINTS synonym for USER_OUTLINE_HINTS view, 2-188
ALL_OUTLINES synonym for USER_OUTLINES view, 2-188
ALL_PART_COL_STATISTICS view, 3-1
ALL_PART_HISTOGRAMS view, 3-2
ALL_PART_INDEXES view, 3-3
ALL_PART_KEY_COLUMNS view, 3-6
ALL_PART_LOBS view, 3-7
ALL_PART_TABLES view, 3-10
ALL_PARTIAL_DROP_TABS view, 3-14
ALL_PENDING_CONV_TABLES view, 3-14

ALL_PLSQL_COLL_TYPES view, 3-15
ALL_PLSQL_OBJECT_SETTINGS view, 3-16
ALL_PLSQL_TYPE_ATTRS view, 3-17
ALL_PLSQL_TYPES view, 3-18
ALL_POLICIES view, 3-19
ALL_POLICY_ATTRIBUTES view, 3-20
ALL_POLICY_CONTEXTS view, 3-21
ALL_POLICY_GROUPS view, 3-21
ALL_PROCEDURES view, 3-22
ALL_PROPAGATION view, 3-23
ALL_QUEUE_SCHEDULES view, 3-25
ALL_QUEUE_SUBSCRIBERS view, 3-27
ALL_QUEUE_TABLES view, 3-28
ALL_QUEUES view, 3-29
ALL_REFRESH view, 3-30
ALL_REFRESH_CHILDREN view, 3-31
ALL_REFRESH_DEPENDENCIES view, 3-32
ALL_REFS view, 3-32
ALL_REGISTERED_MVIEWS view, 3-33
ALL_REGISTRY_BANNERS view, 3-34
ALL_REMOVED_WORKSPACES view, 2-5
ALL_REPCAT view, 2-3
ALL_REPCAT_TEMPLATE_OBJECTS view, 2-3
ALL_REPCAT_TEMPLATE_PARMS view, 2-3
ALL_REPCAT_TEMPLATE_SITES view, 2-3
ALL_REPCAT_USER_AUTHORIZATIONS view, 2-3
ALL_REPCAT_USER_PARM_VALUES view, 2-3
ALL_REPCATLOG view, 2-3
ALL_REPCOLUMN view, 2-4
ALL_REPCOLUMN_GROUP view, 2-4
ALL_REPCONFLICT view, 2-4
ALL_REPDDL view, 2-4
ALL_REPFLAVOR_COLUMNS view, 2-4
ALL_REPFLAVOR_OBJECTS view, 2-4
ALL_REPFLAVORS view, 2-4
ALL_REPGENERATED view, 2-4
ALL_REPGENOBJECTS view, 2-4
ALL_REPGROUP view, 2-4
ALL_REPGROUP_PRIVILEGES view, 2-4
ALL_REPGROUPED_COLUMN view, 2-4
ALL_REPKEY_COLUMNS view, 2-4
ALL_REPL_DBNAME_MAPPING view, 3-34
ALL_REPOBJECT view, 2-4
ALL_REPPARAMETER_COLUMN view, 2-4
ALL_REPPRIORITY view, 2-4
ALL_REPPRIORITY_GROUP view, 2-4
ALL_REPPROP view, 2-4
ALL_REPRESOL_STATS_CONTROL view, 2-4
ALL_REPRESOLUTION view, 2-4
ALL_REPRESOLUTION_METHOD view, 2-4
ALL_REPRESOLUTION_STATISTICS view, 2-4
ALL_REPSCHEMA view, 2-4
ALL_REPSITES view, 2-4
ALL_REWRITE_EQUIVALENCES view, 3-35
ALL_RULE_SET_RULES view, 3-35
ALL_RULE_SETS view, 3-36
ALL_RULES view, 3-37
ALL_SCHEDULER_CHAIN_RULES view, 3-37
ALL_SCHEDULER_CHAIN_STEPS view, 3-38
ALL_SCHEDULER_CHAINS view, 3-40
ALL_SCHEDULER_CREDENTIALS view, 3-40
ALL_SCHEDULER_DB_DESTS view, 3-41
ALL_SCHEDULER_DESTS view, 3-42
ALL_SCHEDULER_EXTERNAL_DESTS view, 3-43
ALL_SCHEDULER_FILE_WATCHERS view, 3-43
ALL_SCHEDULER_GLOBAL_ATTRIBUTE view, 3-44
ALL_SCHEDULER_GROUP_MEMBERS view, 3-45
ALL_SCHEDULER_GROUPS view, 3-45
ALL_SCHEDULER_JOB_ARGS view, 3-46
ALL_SCHEDULER_JOB_CLASSES view, 3-47
ALL_SCHEDULER_JOB_DESTS view, 3-47
ALL_SCHEDULER_JOB_LOG view, 3-49
ALL_SCHEDULER_JOB_RUN_DETAILS view, 3-50
ALL_SCHEDULER_JOBS view, 3-52
ALL_SCHEDULER_NOTIFICATIONS view, 3-57
ALL_SCHEDULER_PROGRAM_ARGS view, 3-58
ALL_SCHEDULER_PROGRAMS view, 3-59
ALL_SCHEDULER_REMOTE_DATABASES view, 3-60
ALL_SCHEDULER_REMOTE_JOBSTATE view, 3-60
ALL_SCHEDULER_RUNNING_CHAINS view, 3-61
ALL_SCHEDULER_RUNNING_JOBS view, 3-63
ALL_SCHEDULER_SCHEDULES view, 3-64
ALL_SCHEDULER_WINDOW_DETAILS view, 3-65
ALL_SCHEDULER_WINDOW_GROUPS view, 3-66
ALL_SCHEDULER_WINDOW_LOG view, 3-67
ALL_SCHEDULER_WINDOWS view, 3-68
ALL_SCHEDULER_WINGROUP_MEMBERS view, 3-69
ALL_SEC_RELEVANT_COLS view, 3-69
ALL_SECONDARY_OBJECTS view, 3-70
ALL_SEQUENCES view, 3-71
ALL_SERVICES view, 3-72
ALL_SOURCE view, 3-75
ALL_SOURCE_AE view, 3-75
ALL_SQL_TRANSLATION_PROFILES view, 3-76
ALL_SQL_TRANSLATIONS view, 3-77
ALL_SQLJ_TYPE_ATTRS view, 3-78
ALL_SQLJ_TYPE_METHODS view, 3-79
ALL_SQLJ_TYPES view, 3-80
ALL_SQLSET view, 3-81
ALL_SQLSET_BINDS view, 3-82
ALL_SQLSET_PLANS view, 3-83
ALL_SQLSET_REFERENCES view, 3-88
ALL_SQLSET_STATEMENTS view, 3-88
ALL_STAT_EXTENSIONS view, 3-91
ALL_STORED_SETTINGS view, 3-91
ALL_STREAMS_COLUMNS view, 3-92
ALL_STREAMS_GLOBAL_RULES view, 3-93
ALL_STREAMS_MESSAGE_CONSUMERS view, 3-94
ALL_STREAMS_MESSAGE_RULES view, 3-95

ALL_STREAMS_NEWLY_SUPPORTED view, *3-95*
ALL_STREAMS_RULES view, *3-96*
ALL_STREAMS_SCHEMA_RULES view, *3-98*
ALL_STREAMS_TABLE_RULES view, *3-99*
ALL_STREAMS_TRANSFORM_FUNCTION view, *3-101*
ALL_STREAMS_UNSUPPORTED view, *3-101*
ALL_SUBPART_COL_STATISTICS view, *3-102*
ALL_SUBPART_HISTOGRAMS view, *3-104*
ALL_SUBPART_KEY_COLUMNS view, *3-105*
ALL_SUBPARTITION_TEMPLATES view, *3-105*
ALL_SUMDELTA view, *3-106*
ALL_SYNC_CAPTURE view, *3-106*
ALL_SYNC_CAPTURE_PREPARED_TABS view, *3-107*
ALL_SYNC_CAPTURE_TABLES view, *3-108*
ALL_SYNONYMS view, *3-108*
ALL_TAB_COL_STATISTICS view, *3-109*
ALL_TAB_COLS view, *3-111*
ALL_TAB_COLUMNS view, *3-114*
ALL_TAB_COMMENTS view, *3-117*
ALL_TAB_HISTGRM_PENDING_STATS view, *3-118*
ALL_TAB_HISTOGRAMS view
  ALL_HISTOGRAMS synonym, *2-107*
ALL_TAB_IDENTITY_COLS view, *3-120*
ALL_TAB_MODIFICATIONS view, *3-121*
ALL_TAB_PARTITIONS view, *3-122*
ALL_TAB_PENDING_STATS view, *3-127*
ALL_TAB_PRIVS view, *3-128*
ALL_TAB_PRIVS_MADE view, *3-129*
ALL_TAB_PRIVS_RECD view, *3-129*
ALL_TAB_STAT_PREFS view, *3-130*
ALL_TAB_STATISTICS view, *3-131*
ALL_TAB_STATS_HISTORY view, *3-133*
ALL_TAB_SUBPARTITIONS view, *3-133*
ALL_TABLES view, *3-137*
ALL_TRANSFORMATIONS view, *3-142*
ALL_TRIGGER_COLS view, *3-142*
ALL_TRIGGER_ORDERING view, *3-143*
ALL_TRIGGERS view, *3-144*
ALL_TSTZ_TAB_COLS view, *3-146*
ALL_TSTZ_TABLES view, *3-147*
ALL_TYPE_ATTRS view, *3-147*
ALL_TYPE_METHODS view, *3-148*
ALL_TYPE_VERSIONS view, *3-149*
ALL_TYPES view, *3-150*
ALL_UNUSED_COL_TABS view, *3-151*
ALL_UPDATABLE_COLUMNS view, *3-152*
ALL_USERS view, *3-152*
ALL_USTATS view, *3-153*
ALL_VARRAYS view, *3-154*
ALL_VERSION_HVIEW view, *2-5*
ALL_VIEWS view, *3-155*
ALL_VIEWS_AE view, *3-156*
ALL_WARNING_SETTINGS view, *3-157*
ALL_WM_CONS_COLUMNS view, *2-5*
ALL_WM_CONSTRAINTS view, *2-5*
ALL_WM_IND_COLUMNS view, *2-5*
ALL_WM_IND_EXPRESSIONS view, *2-5*
ALL_WM_LOCKED_TABLES view, *2-5*
ALL_WM_MODIFIED_TABLES view, *2-5*
ALL_WM_RIC_INFO view, *2-5*
ALL_WM_TAB_TRIGGERS view, *2-5*
ALL_WM_VERSIONED_TABLES view, *2-5*
ALL_WM_VT_ERRORS view, *2-5*
ALL_WORKSPACE_PRIVS view, *2-5*
ALL_WORKSPACE_SAVEPOINTS view, *2-5*
ALL_WORKSPACES view, *2-6*
ALL_XML_INDEXES view, *3-158*
ALL_XML_NESTED_TABLES view, *3-159*
ALL_XML_OUT_OF_LINE_TABLES view, *3-160*
ALL_XML_SCHEMA_ATTRIBUTES view, *3-160*
ALL_XML_SCHEMA_COMPLEX_TYPES view, *3-161*
ALL_XML_SCHEMA_ELEMENTS view, *3-162*
ALL_XML_SCHEMA_NAMESPACES view, *3-163*
ALL_XML_SCHEMA_SIMPLE_TYPES view, *3-164*
ALL_XML_SCHEMA_SUBSTGRP_HEAD view, *3-165*
ALL_XML_SCHEMA_SUBSTGRP_MBRS view, *3-165*
ALL_XML_SCHEMAS view, *3-166*
ALL_XML_TAB_COLS view, *3-167*
ALL_XML_TABLES view, *3-168*
ALL_XML_VIEW_COLS view, *3-169*
ALL_XML_VIEWS view, *3-170*
ALL_XSTREAM_ADMINISTRATOR view, *3-170*
ALL_XSTREAM_INBOUND view, *3-171*
ALL_XSTREAM_INBOUND_PROGRESS view, *3-172*
ALL_XSTREAM_OUT_SUPPORT_MODE view, *3-173*
ALL_XSTREAM_OUTBOUND view, *3-174*
ALL_XSTREAM_OUTBOUND_PROGRESS view, *3-175*
ALL_XSTREAM_RULES view, *3-176*
ALL_XSTREAM_TRANSFORMATIONS view, *3-177*
ALL_ZONEMAP_MEASURES view, *3-179*
ALL_ZONEMAPS view, *3-179*
ALLOCATE EXTENT clause
  of ALTER TABLE
    instance number, *1-111*
ALTER DATABASE statement
  ADD LOGFILE, *1-242*
  ENABLE THREAD, *1-242*
  THREAD, *1-242*
ALTER TABLE statement
  allocating extents, *1-111*
AQ_TM_PROCESSES initialization parameter, *1-23*
ARCHIVE LOG START statement
  automatic archiving, *1-125*
ARCHIVE_LAG_TARGET initialization parameter, *1-24*
archived redo logs
  destination file, *1-124*
  storage device, *1-124*, *1-131*
ARCHIVELOG mode

ARCHIVELOG mode *(continued)*
archiving destination, *1-124*, *1-131*
ASM_DISKGROUPS initialization parameter, *1-25*
ASM_DISKSTRING initialization parameter, *1-25*
ASM_POWER_LIMIT initialization parameter, *1-26*
ASM_PREFERRED_READ_FAILURE_GROUPS initialization parameter, *1-27*
AUDIT_ACTIONS table, *3-181*
AUDIT_FILE_DEST initialization parameter, *1-27*
AUDIT_SYS_OPERATIONS initialization parameter, *1-28*
AUDIT_SYSLOG_LEVEL initialization parameter, *1-29*
AUDIT_TRAIL initialization parameter, *1-30*
AUDIT_UNIFIED_CONTEXTS view, *3-182*
AUDIT_UNIFIED_ENABLED_POLICIES view, *3-182*
AUDIT_UNIFIED_POLICIES view, *3-183*
AUDIT_UNIFIED_POLICY_COMMENTS view, *3-184*
AUDITABLE_SYSTEM_ACTIONS view, *3-184*
authenticating remote clients, *1-208*
authenticating users, *1-176*
AWR_SNAPSHOT_TIME_OFFSET initialization parameter, *1-33*

## B

BACKGROUND_CORE_DUMP initialization parameter, *1-33*
BACKGROUND_DUMP_DEST initialization parameter, *1-34*
BACKUP_TAPE_IO_SLAVES initialization parameter, *1-35*
BITMAP_MERGE_AREA_SIZE initialization parameter, *1-35*
BLANK_TRIMMING initialization parameter, *1-36*
blocks
redo log, *1-138*
size, *1-54*, *1-59*, *1-69*

## C

cache
dictionary, *8-160–8-162*
statistics, *1-138*
CAT synonym for USER_CATALOG view, *3-186*
CATALOG view, *3-186*
CATALOG.SQL script
creating V$ views, *7-1*
CATBLOCK.SQL script, *B-2*
CATCLUST.SQL script, *B-2*
CATHS.SQL script, *B-2*
CATJAVA.SQL script, *B-6*
CATNOCLUST.SQL script, *B-5*
CATNOJAV.SQL script, *B-5*
CATNOPRT.SQL script, *B-5*
CATNOSVM.SQL script, *B-5*
CATNSNMP.SQL script, *B-6*
CATPROC.SQL script, *B-2*
CATWRR.SQL script, *B-3*
CDB_* views, *2-2*
CDBs, *1-28*
CHAINED_ROWS table, *3-186*
characters
numeric group separators, *1-157*
checkpoints
checkpoint interval, *1-138*
statistics, *1-138*
CIRCUITS initialization parameter, *1-37*
CLIENT_RESULT_CACHE_LAG initialization parameter, *1-37*
CLIENT_RESULT_CACHE_SIZE initialization parameter, *1-38*
CLIENT_RESULT_CACHE_STATS$ view, *3-186*
CLONEDB initialization parameter, *1-38*
CLU synonym for USER_CLUSTERS view, *3-187*
CLUSTER_DATABASE initialization parameter, *1-39*
CLUSTER_DATABASE_INSTANCES initialization parameter, *1-39*
CLUSTER_INTERCONNECTS initialization parameter, *1-40*
COL view, *3-187*
COLS synonym for USER_TAB_COLUMNS view, *3-187*
commit wait/nowait performed, *E-4*
COMMIT_LOGGING initialization parameter, *1-41*
COMMIT_POINT_STRENGTH initialization parameter, *1-41*
COMMIT_WAIT initialization parameter, *1-42*
COMMIT_WRITE initialization parameter, *1-42*
COMMON_USER_PREFIX initialization parameter, *1-43*
COMPATIBLE initialization parameter, *1-44*
CONNECTION_BROKERS initialization parameter, *1-45*
contention
block-level, *9-155*
control files
names, *1-47*
specifying, *1-47*
CONTROL_FILE_RECORD_KEEP_TIME initialization parameter, *1-46*
CONTROL_FILES initialization parameter, *1-47*
CONTROL_MANAGEMENT_PACK_ACCESS initialization parameter, *1-48*
CORE_DUMP_DEST initialization parameter, *1-48*
cost-based optimization
favoring IN-list iterators, *1-171*
CPU_COUNT initialization parameter, *1-49*
CREATE TABLE statement
FREELIST GROUPS clause, *1-111*
CREATE_BITMAP_AREA_SIZE initialization parameter, *1-50*
CREATE_STORED_OUTLINES initialization parameter, *1-50*

creating
a database
setting block size, 1-54, 1-59, 1-69
currency
international currency symbol, 1-154
CURSOR_BIND_CAPTURE_DESTINATION
initialization parameter, 1-51
CURSOR_SHARING initialization parameter, 1-52
CURSOR_SPACE_FOR_TIME initialization parameter, 1-53
cursors
OPEN_CURSORS initialization parameter, 1-164
shared pool, 1-227

## D

data blocks
reading multiple, 1-63
size of, 1-54, 1-59, 1-69
data dictionary
cache, 8-160–8-162
scripts, B-2
SQL scripts, B-1
tables, 2-2
views
overview, 2-2
user views, 2-1
database writer process (DBWR)
checkpoint, 1-138
DATABASE_EXPORT_OBJECTS view, 3-187
DATABASE_PROPERTIES view, 3-188
databases
database limits, A-1
default language, 1-154
resource limits, 1-209, 1-212
datafiles
maximum number, 1-65
parameter, 1-65
dates
language used for, 1-152, 1-160, 1-161
setting SYSDATE, 1-95
TO_CHAR function, 1-151
TO_DATE function, 1-151
DB_BIG_TABLE_CACHE_PERCENT_TARGET
initialization parameter, 1-54
DB_BLOCK_BUFFERS initialization parameter, 1-56
DB_BLOCK_CHECKING initialization parameter, 1-57
DB_BLOCK_CHECKSUM initialization parameter, 1-58
DB_BLOCK_SIZE initialization parameter, 1-59
DB_CACHE_ADVICE initialization parameter, 1-59
DB_CACHE_SIZE initialization parameter, 1-60
DB_CREATE_FILE_DEST initialization parameter, 1-61
DB_CREATE_ONLINE_LOG_DEST_n initialization parameter, 1-62
DB_DOMAIN initialization parameter, 1-62
DB_FILE_MULTIBLOCK_READ_COUNT
initialization parameter, 1-63
DB_FILE_NAME_CONVERT initialization parameter, 1-64
DB_FILES initialization parameter, 1-65
DB_FLASH_CACHE_FILE initialization parameter, 1-66
DB_FLASH_CACHE_SIZE initialization parameter, 1-66
DB_FLASHBACK_RETENTION_TARGET
initialization parameter, 1-67
DB_INDEX_COMPRESSION_INHERITANCE
initialization parameter, 1-68
DB_KEEP_CACHE_SIZE initialization parameter, 1-68
DB_LOST_WRITE_PROTECT initialization parameter, 1-69
DB_NAME initialization parameter, 1-70
DB_nK_CACHE_SIZE initialization parameter, 1-54
DB_RECOVERY_FILE_DEST initialization parameter, 1-70
DB_RECOVERY_FILE_DEST_SIZE initialization parameter, 1-71
DB_RECYCLE_CACHE_SIZE initialization parameter, 1-72
DB_SECUREFILE initialization parameter, 1-72
DB_ULTRA_SAFE initialization parameter, 1-74
DB_UNIQUE_NAME initialization parameter, 1-74
DB_UNRECOVERABLE_SCN_TRACKING
initialization parameter, 1-75
DB_WRITER_PROCESSES initialization parameter, 1-76
DBA_2PC_NEIGHBORS view, 4-1
DBA_2PC_PENDING view, 4-1
DBA_ACL_NAME_MAP view, 4-2
DBA_ADDM_FDG_BREAKDOWN view, 4-2
DBA_ADDM_FINDINGS view, 4-3
DBA_ADDM_INSTANCES view, 4-5
DBA_ADDM_SYSTEM_DIRECTIVES view, 4-6
DBA_ADDM_TASK_DIRECTIVES view, 4-6
DBA_ADDM_TASKS view, 4-7
DBA_ADVISOR_ACTIONS view, 4-9
DBA_ADVISOR_COMMANDS view, 4-11
DBA_ADVISOR_DEF_PARAMETERS view, 4-11
DBA_ADVISOR_DEFINITIONS view, 4-12
DBA_ADVISOR_DIR_DEFINITIONS view, 4-13
DBA_ADVISOR_DIR_INSTANCES view, 4-14
DBA_ADVISOR_DIR_TASK_INST view, 4-14
DBA_ADVISOR_EXEC_PARAMETERS view, 4-15
DBA_ADVISOR_EXECUTION_TYPES view, 4-16
DBA_ADVISOR_EXECUTIONS view, 4-16
DBA_ADVISOR_FDG_BREAKDOWN view, 4-17
DBA_ADVISOR_FINDING_NAMES view, 4-18
DBA_ADVISOR_FINDINGS view, 4-18
DBA_ADVISOR_JOURNAL view, 4-19
DBA_ADVISOR_LOG view, 4-20

DBA_ADVISOR_OBJECT_TYPES view, *4-21*
DBA_ADVISOR_OBJECTS view, *4-22*
DBA_ADVISOR_PARAMETERS view, *4-23*
DBA_ADVISOR_RATIONALE view, *4-25*
DBA_ADVISOR_RECOMMENDATIONS view, *4-26*
DBA_ADVISOR_SQLA_REC_SUM view, *4-27*
DBA_ADVISOR_SQLA_TABLES view, *4-28*
DBA_ADVISOR_SQLA_WK_MAP view, *4-28*
DBA_ADVISOR_SQLA_WK_STMTS view, *4-29*
DBA_ADVISOR_SQLPLANS view, *4-31*
DBA_ADVISOR_SQLSTATS view, *4-34*
DBA_ADVISOR_SQLW_JOURNAL view, *4-36*
DBA_ADVISOR_SQLW_PARAMETERS view, *4-36*
DBA_ADVISOR_SQLW_STMTS view, *4-37*
DBA_ADVISOR_SQLW_SUM view, *4-39*
DBA_ADVISOR_SQLW_TABLES view, *4-40*
DBA_ADVISOR_SQLW_TEMPLATES view, *4-40*
DBA_ADVISOR_TASKS view, *4-41*
DBA_ADVISOR_TEMPLATES view, *4-43*
DBA_ADVISOR_USAGE view, *4-43*
DBA_ALERT_HISTORY view, *4-44*
DBA_ALERT_HISTORY_DETAIL view, *4-45*
DBA_ALL_TABLES view, *4-47*
DBA_APPLICATION_ROLES view, *4-47*
DBA_APPLY view, *4-47*
DBA_APPLY_CHANGE_HANDLERS view, *4-47*
DBA_APPLY_CONFLICT_COLUMNS view, *4-48*
DBA_APPLY_DML_HANDLERS view, *4-48*
DBA_APPLY_ENQUEUE view, *4-48*
DBA_APPLY_ERROR view, *4-48*
DBA_APPLY_ERROR_MESSAGES view, *4-48*
DBA_APPLY_EXECUTE view, *4-49*
DBA_APPLY_INSTANTIATED_GLOBAL view, *4-49*
DBA_APPLY_INSTANTIATED_OBJECTS view, *4-49*
DBA_APPLY_INSTANTIATED_SCHEMAS view, *4-49*
DBA_APPLY_KEY_COLUMNS view, *4-50*
DBA_APPLY_OBJECT_DEPENDENCIES view, *4-50*
DBA_APPLY_PARAMETERS view, *4-50*
DBA_APPLY_PROGRESS view, *4-50*
DBA_APPLY_SPILL_TXN view, *4-51*
DBA_APPLY_TABLE_COLUMNS view, *4-51*
DBA_APPLY_VALUE_DEPENDENCIES view, *4-51*
DBA_AQ_AGENT_PRIVS view, *4-52*
DBA_AQ_AGENTS view, *4-52*
DBA_ARGUMENTS view, *4-52*
DBA_ASSEMBLIES view, *4-53*
DBA_ASSOCIATIONS view, *4-53*
DBA_ATTRIBUTE_TRANSFORMATIONS view, *4-53*
DBA_AUDIT_EXISTS view, *4-53*
DBA_AUDIT_MGMT_CLEAN_EVENTS view, *4-55*
DBA_AUDIT_MGMT_CLEANUP_JOBS view, *4-56*
DBA_AUDIT_MGMT_CONFIG_PARAMS view, *4-58*
DBA_AUDIT_MGMT_LAST_ARCH_TS view, *4-59*
DBA_AUDIT_OBJECT view, *4-60*
DBA_AUDIT_POLICIES view, *4-62*
DBA_AUDIT_POLICY_COLUMNS view, *4-63*
DBA_AUDIT_SESSION view, *4-63*
DBA_AUDIT_STATEMENT view, *4-65*
DBA_AUDIT_TRAIL view, *4-67*
DBA_AUTO_SEGADV_CTL view, *4-70*
DBA_AUTO_SEGADV_SUMMARY view, *4-71*
DBA_AUTOTASK_CLIENT view, *4-71*
DBA_AUTOTASK_CLIENT_HISTORY view, *4-73*
DBA_AUTOTASK_CLIENT_JOB view, *4-73*
DBA_AUTOTASK_JOB_HISTORY view, *4-74*
DBA_AUTOTASK_OPERATION view, *4-75*
DBA_AUTOTASK_SCHEDULE view, *4-75*
DBA_AUTOTASK_STATUS view, *4-76*
DBA_AUTOTASK_TASK view, *4-76*
DBA_AUTOTASK_WINDOW_CLIENTS view, *4-80*
DBA_AUTOTASK_WINDOW_HISTORY view, *4-80*
DBA_AW_PS view, *4-81*
DBA_AWS view, *4-81*
DBA_BASE_TABLE_MVIEWS view, *4-81*
DBA_BLOCKERS view, *4-81*
DBA_CAPTURE view, *4-82*
DBA_CAPTURE_EXTRA_ATTRIBUTES view, *4-82*
DBA_CAPTURE_PARAMETERS view, *4-82*
DBA_CAPTURE_PREPARED_DATABASE view, *4-82*
DBA_CAPTURE_PREPARED_SCHEMAS view, *4-82*
DBA_CAPTURE_PREPARED_TABLES view, *4-83*
DBA_CATALOG view, *4-83*
DBA_CDB_RSRC_PLAN_DIRECTIVES view, *4-83*
DBA_CDB_RSRC_PLANS view, *4-84*
DBA_CHANGE_NOTIFICATION_REGS view, *4-84*
DBA_CLU_COLUMNS view, *4-85*
DBA_CLUSTER_HASH_EXPRESSIONS view, *4-85*
DBA_CLUSTERING_DIMENSIONS view, *4-85*
DBA_CLUSTERING_JOINS view, *4-86*
DBA_CLUSTERING_KEYS view, *4-86*
DBA_CLUSTERING_TABLES view, *4-86*
DBA_CLUSTERS view, *4-87*
DBA_CODE_ROLE_PRIVS view, *4-87*
DBA_COL_COMMENTS view, *4-87*
DBA_COL_PENDING_STATS view, *4-87*
DBA_COL_PRIVS view, *4-87*
DBA_COLL_TYPES view, *4-88*
DBA_COMMON_AUDIT_TRAIL view, *4-88*
DBA_COMPARISON view, *4-92*
DBA_COMPARISON_COLUMNS view, *4-93*
DBA_COMPARISON_ROW_DIF view, *4-93*
DBA_COMPARISON_SCAN view, *4-94*
DBA_COMPARISON_SCAN_VALUES view, *4-95*
DBA_CONNECT_ROLE_GRANTEES view, *4-95*
DBA_CONS_COLUMNS view, *4-96*
DBA_CONS_OBJ_COLUMNS view, *4-96*
DBA_CONSTRAINTS view, *4-96*
DBA_CONTAINER_DATA view, *4-96*
DBA_CONTEXT view, *4-97*
DBA_CPOOL_INFO view, *4-97*

DBA_CPU_USAGE_STATISTICS view, *4-98*
DBA_CQ_NOTIFICATION_QUERIES view, *4-99*
DBA_CREDENTIALS view, *4-99*
DBA_CUBE_ATTR_VISIBILITY view, *4-99*
DBA_CUBE_ATTRIBUTES view, *4-99*
DBA_CUBE_BUILD_PROCESSES view, *4-100*
DBA_CUBE_CALCULATED_MEMBERS view, *4-100*
DBA_CUBE_DIM_LEVELS view, *4-100*
DBA_CUBE_DIM_MODELS view, *4-100*
DBA_CUBE_DIM_VIEW_COLUMNS view, *4-100*
DBA_CUBE_DIM_VIEWS view, *4-101*
DBA_CUBE_DIMENSIONALITY view, *4-101*
DBA_CUBE_DIMENSIONS view, *4-101*
DBA_CUBE_HIER_LEVELS view, *4-101*
DBA_CUBE_HIER_VIEW_COLUMNS view, *4-101*
DBA_CUBE_HIER_VIEWS view, *4-102*
DBA_CUBE_HIERARCHIES view, *4-102*
DBA_CUBE_MEASURES view, *4-102*
DBA_CUBE_NAMED_BUILD_SPECS view, *4-102*
DBA_CUBE_SUB_PARTITION_LEVELS view, *4-102*
DBA_CUBE_VIEW_COLUMNS view, *4-103*
DBA_CUBE_VIEWS view, *4-103*
DBA_CUBES view, *4-103*
DBA_DATA_FILES view, *4-103*
DBA_DATAPUMP_JOBS view, *4-104*
DBA_DATAPUMP_SESSIONS view, *4-105*
DBA_DB_LINKS view, *4-105*
DBA_DBFS_HS view, *4-105*
DBA_DBFS_HS_COMMANDS view, *4-106*
DBA_DBFS_HS_FIXED_PROPERTIES view, *4-106*
DBA_DBFS_HS_PROPERTIES view, *4-107*
DBA_DDL_LOCKS view, *4-107*
DBA_DEPENDENCIES view, *4-108*
DBA_DIGEST_VERIFIERS view, *4-108*
DBA_DIM_ATTRIBUTES view, *4-108*
DBA_DIM_CHILD_OF view, *4-109*
DBA_DIM_HIERARCHIES view, *4-109*
DBA_DIM_JOIN_KEY view, *4-109*
DBA_DIM_LEVEL_KEY view, *4-109*
DBA_DIM_LEVELS view, *4-109*
DBA_DIMENSIONS view, *4-109*
DBA_DIRECTORIES view, *4-109*
DBA_DISCOVERY_SOURCE view, *4-109*
DBA_DML_LOCKS view, *4-110*
DBA_DMT_FREE_SPACE view, *4-110*
DBA_DMT_USED_EXTENTS view, *4-111*
DBA_EDITION_COMMENTS view, *4-111*
DBA_EDITIONED_TYPES view, *4-111*
DBA_EDITIONING_VIEW_COLS view, *4-112*
DBA_EDITIONING_VIEW_COLS_AE view, *4-112*
DBA_EDITIONING_VIEWS view, *4-112*
DBA_EDITIONING_VIEWS_AE view, *4-112*
DBA_EDITIONS view, *4-113*
DBA_ENABLED_AGGREGATIONS view, *4-113*
DBA_ENABLED_TRACES view, *4-113*
DBA_ENCRYPTED_COLUMNS view, *4-114*
DBA_EPG_DAD_AUTHORIZATION view, *4-114*
DBA_ERROR_TRANSLATIONS view, *4-114*
DBA_ERRORS view, *4-115*
DBA_ERRORS_AE view, *4-115*
DBA_EVALUATION_CONTEXT_TABLES view, *4-115*
DBA_EVALUATION_CONTEXT_VARS view, *4-115*
DBA_EVALUATION_CONTEXTS view, *4-115*
DBA_EXP_FILES view, *4-116*
DBA_EXP_OBJECTS view, *4-116*
DBA_EXP_VERSION view, *4-116*
DBA_EXTENTS view, *4-116*
DBA_EXTERNAL_LOCATIONS view, *4-117*
DBA_EXTERNAL_TABLES view, *4-117*
DBA_FEATURE_USAGE_STATISTICS view, *4-118*
DBA_FGA_AUDIT_TRAIL view, *4-118*
DBA_FILE_GROUP_EXPORT_INFO view, *4-120*
DBA_FILE_GROUP_FILES view, *4-120*
DBA_FILE_GROUP_TABLES view, *4-120*
DBA_FILE_GROUP_TABLESPACES view, *4-121*
DBA_FILE_GROUP_VERSIONS view, *4-121*
DBA_FILE_GROUPS view, *4-121*
DBA_FLASHBACK_ARCHIVE view, *4-121*
DBA_FLASHBACK_ARCHIVE_TABLES view, *4-122*
DBA_FLASHBACK_ARCHIVE_TS view, *4-122*
DBA_FLASHBACK_TXN_REPORT view, *4-122*
DBA_FLASHBACK_TXN_STATE view, *4-123*
DBA_FREE_SPACE view, *4-124*
DBA_FREE_SPACE_COALESCED view, *4-124*
DBA_GG_INBOUND_PROGRESS view, *4-125*
DBA_GLOBAL_CONTEXT view, *4-125*
DBA_GOLDENGATE_INBOUND view, *4-126*
DBA_GOLDENGATE_PRIVILEGES view, *4-126*
DBA_GOLDENGATE_RULES view, *4-126*
DBA_GOLDENGATE_SUPPORT_MODE view, *4-126*
DBA_HEAT_MAP_SEG_HISTOGRAM view, *4-127*
DBA_HEAT_MAP_SEGMENT view, *4-127*
DBA_HEATMAP_TOP_OBJECTS view, *4-127*
DBA_HEATMAP_TOP_TABLESPACES view, *4-128*
DBA_HIGH_WATER_MARK_STATISTICS view, *4-129*
DBA_HIST_ACTIVE_SESS_HISTORY view, *4-130*
DBA_HIST_APPLY_SUMMARY view, *4-136*
DBA_HIST_ASH_SNAPSHOT view, *4-138*
DBA_HIST_BASELINE view, *4-139*
DBA_HIST_BASELINE_DETAILS view, *4-140*
DBA_HIST_BASELINE_METADATA view, *4-141*
DBA_HIST_BASELINE_TEMPLATE view, *4-142*
DBA_HIST_BG_EVENT_SUMMARY view, *4-143*
DBA_HIST_BUFFER_POOL_STAT view, *4-144*
DBA_HIST_BUFFERED_QUEUES view, *4-145*
DBA_HIST_BUFFERED_SUBSCRIBERS view, *4-147*
DBA_HIST_CAPTURE view, *4-148*
DBA_HIST_CLUSTER_INTERCON view, *4-150*
DBA_HIST_COLORED_SQL view, *4-150*
DBA_HIST_COMP_IOSTAT view, *4-151*

DBA_HIST_CR_BLOCK_SERVER view, *4-152*
DBA_HIST_CURRENT_BLOCK_SERVER view, *4-153*
DBA_HIST_DATABASE_INSTANCE view, *4-154*
DBA_HIST_DATAFILE view, *4-155*
DBA_HIST_DB_CACHE_ADVICE view, *4-156*
DBA_HIST_DISPATCHER view, *4-157*
DBA_HIST_DLM_MISC view, *4-158*
DBA_HIST_DYN_REMASTER_STATS view, *4-158*
DBA_HIST_ENQUEUE_STAT view, *4-160*
DBA_HIST_EVENT_HISTOGRAM view, *4-161*
DBA_HIST_EVENT_NAME view, *4-161*
DBA_HIST_FILEMETRIC_HISTORY view, *4-162*
DBA_HIST_FILESTATXS view, *4-163*
DBA_HIST_IC_CLIENT_STATS view, *4-164*
DBA_HIST_IC_DEVICE_STATS view, *4-165*
DBA_HIST_IM_SEG_STAT view, *4-166*
DBA_HIST_IM_SEG_STAT_OBJ view, *4-167*
DBA_HIST_INST_CACHE_TRANSFER view, *4-168*
DBA_HIST_INSTANCE_RECOVERY view, *4-170*
DBA_HIST_INTERCONNECT_PINGS view, *4-172*
DBA_HIST_IOSTAT_DETAIL view, *4-174*
DBA_HIST_IOSTAT_FILETYPE view, *4-175*
DBA_HIST_IOSTAT_FILETYPE_NAME view, *4-176*
DBA_HIST_IOSTAT_FUNCTION view, *4-177*
DBA_HIST_IOSTAT_FUNCTION_NAME view, *4-178*
DBA_HIST_JAVA_POOL_ADVICE view, *4-178*
DBA_HIST_LATCH view, *5-1*
DBA_HIST_LATCH_CHILDREN view, *5-2*
DBA_HIST_LATCH_MISSES_SUMMARY view, *5-3*
DBA_HIST_LATCH_NAME view, *5-4*
DBA_HIST_LATCH_PARENT view, *5-5*
DBA_HIST_LIBRARYCACHE view, *5-6*
DBA_HIST_LOG view, *5-7*
DBA_HIST_MEM_DYNAMIC_COMP view, *5-8*
DBA_HIST_MEMORY_RESIZE_OPS view, *5-9*
DBA_HIST_MEMORY_TARGET_ADVICE view, *5-11*
DBA_HIST_METRIC_NAME view, *5-12*
DBA_HIST_MTTR_TARGET_ADVICE view, *5-12*
DBA_HIST_MUTEX_SLEEP view, *5-13*
DBA_HIST_OPTIMIZER_ENV view, *5-14*
DBA_HIST_OSSTAT view, *5-15*
DBA_HIST_OSSTAT_NAME view, *5-15*
DBA_HIST_PARAMETER view, *5-16*
DBA_HIST_PARAMETER_NAME view, *5-17*
DBA_HIST_PDB_INSTANCE view, *5-17*
DBA_HIST_PERSISTENT_QMN_CACHE view, *5-18*
DBA_HIST_PERSISTENT_QUEUES view, *5-20*
DBA_HIST_PERSISTENT_SUBS view, *5-21*
DBA_HIST_PGA_TARGET_ADVICE view, *5-23*
DBA_HIST_PGASTAT view, *5-24*
DBA_HIST_PLAN_OPERATION_NAME view, *5-25*
DBA_HIST_PLAN_OPTION_NAME view, *5-26*
DBA_HIST_PROCESS_MEM_SUMMARY view, *5-26*
DBA_HIST_REPLICATION_TBL_STATS view, *5-27*
DBA_HIST_REPLICATION_TXN_STATS view, *5-28*
DBA_HIST_REPORTS view, *5-29*
DBA_HIST_REPORTS_CONTROL view, *5-30*
DBA_HIST_REPORTS_DETAILS view, *5-31*
DBA_HIST_REPORTS_TIMEBANDS view, *5-32*
DBA_HIST_RESOURCE_LIMIT view, *5-33*
DBA_HIST_ROWCACHE_SUMMARY view, *5-34*
DBA_HIST_RSRC_CONSUMER_GROUP view, *5-35*
DBA_HIST_RSRC_PLAN view, *5-38*
DBA_HIST_RULE_SET view, *5-39*
DBA_HIST_SEG_STAT view, *5-40*
DBA_HIST_SEG_STAT_OBJ view, *5-43*
DBA_HIST_SERVICE_NAME view, *5-44*
DBA_HIST_SERVICE_STAT view, *5-44*
DBA_HIST_SERVICE_WAIT_CLASS view, *5-45*
DBA_HIST_SESS_SGA_STATS view, *5-46*
DBA_HIST_SESS_TIME_STATS view, *5-47*
DBA_HIST_SESSMETRIC_HISTORY view, *5-47*
DBA_HIST_SGA view, *5-48*
DBA_HIST_SGA_TARGET_ADVICE view, *5-49*
DBA_HIST_SGASTAT view, *5-50*
DBA_HIST_SHARED_POOL_ADVICE view, *5-50*
DBA_HIST_SHARED_SERVER_SUMMARY view, *5-52*
DBA_HIST_SNAP_ERROR view, *5-53*
DBA_HIST_SNAPSHOT view, *5-54*
DBA_HIST_SQL_BIND_METADATA view, *5-55*
DBA_HIST_SQL_PLAN view, *5-56*
DBA_HIST_SQL_SUMMARY view, *5-59*
DBA_HIST_SQL_WORKAREA_HSTGRM view, *5-59*
DBA_HIST_SQLBIND view, *5-60*
DBA_HIST_SQLCOMMAND_NAME view, *5-62*
DBA_HIST_SQLSTAT view, *5-62*
DBA_HIST_SQLTEXT view, *5-68*
DBA_HIST_STAT_NAME view, *5-68*
DBA_HIST_STREAMS_APPLY_SUM view, *5-69*
DBA_HIST_STREAMS_CAPTURE view, *5-71*
DBA_HIST_STREAMS_POOL_ADVICE view, *5-72*
DBA_HIST_SYS_TIME_MODEL view, *5-72*
DBA_HIST_SYSMETRIC_HISTORY view, *5-73*
DBA_HIST_SYSMETRIC_SUMMARY view, *5-74*
DBA_HIST_SYSSTAT view, *5-75*
DBA_HIST_SYSTEM_EVENT view, *5-75*
DBA_HIST_TABLESPACE view, *5-76*
DBA_HIST_TABLESPACE_STAT view, *5-77*
DBA_HIST_TBSPC_SPACE_USAGE view, *5-78*
DBA_HIST_TEMPFILE view, *5-79*
DBA_HIST_TEMPSTATXS view, *5-79*
DBA_HIST_THREAD view, *5-81*
DBA_HIST_TOPLEVELCALL_NAME view, *5-81*
DBA_HIST_UNDOSTAT view, *5-82*
DBA_HIST_WAITCLASSMET_HISTORY view, *5-84*
DBA_HIST_WAITSTAT view, *5-85*
DBA_HIST_WR_CONTROL view, *5-86*
DBA_HISTOGRAMS synonym for DBA_TAB_HISTOGRAMS, *5-86*
DBA_HOST_ACES view, *5-86*

DBA_HOST_ACLS view, *5-87*
DBA_IDENTIFIERS view, *5-87*
DBA_ILMDATAMOVEMENTPOLICIES view, *5-88*
DBA_ILMEVALUATIONDETAILS view, *5-89*
DBA_ILMOBJECTS view, *5-90*
DBA_ILMPARAMETERS view, *5-91*
DBA_ILMPOLICIES view, *5-92*
DBA_ILMRESULTS view, *5-93*
DBA_ILMTASKS view, *5-93*
DBA_IND_COLUMNS view, *5-94*
DBA_IND_EXPRESSIONS view, *5-94*
DBA_IND_PARTITIONS view, *5-94*
DBA_IND_PENDING_STATS view, *5-95*
DBA_IND_STATISTICS view, *5-95*
DBA_IND_SUBPARTITIONS view, *5-95*
DBA_INDEXES view
  collecting statistics for, *5-95*
DBA_INDEXTYPE_ARRAYTYPES view, *5-95*
DBA_INDEXTYPE_COMMENTS view, *5-95*
DBA_INDEXTYPE_OPERATORS view, *5-96*
DBA_INDEXTYPES view, *5-96*
DBA_INTERNAL_TRIGGERS view, *5-96*
DBA_JAVA_ARGUMENTS view, *5-96*
DBA_JAVA_CLASSES view, *5-96*
DBA_JAVA_COMPILER_OPTIONS view, *5-96*
DBA_JAVA_DERIVATIONS view, *5-97*
DBA_JAVA_FIELDS view, *5-97*
DBA_JAVA_IMPLEMENTS view, *5-97*
DBA_JAVA_INNERS view, *5-97*
DBA_JAVA_LAYOUTS view, *5-97*
DBA_JAVA_METHODS view, *5-98*
DBA_JAVA_NCOMPS view, *5-98*
DBA_JAVA_POLICY view, *5-98*
DBA_JAVA_RESOLVERS view, *5-99*
DBA_JAVA_THROWS view, *5-99*
DBA_JOBS view, *5-99*
DBA_JOBS_RUNNING view, *5-100*
DBA_JOIN_IND_COLUMNS view, *5-101*
DBA_JSON_COLUMNS view, *5-101*
DBA_KGLLOCK view, *5-101*
DBA_LIBRARIES view, *5-102*
DBA_LMT_FREE_SPACE view, *5-102*
DBA_LMT_USED_EXTENTS view, *5-102*
DBA_LOB_PARTITIONS view, *5-103*
DBA_LOB_SUBPARTITIONS view, *5-103*
DBA_LOB_TEMPLATES view, *5-103*
DBA_LOBS view, *5-103*
DBA_LOCK view
  DBA_LOCKS synonym, *5-104*
DBA_LOCK_INTERNAL view, *5-104*
DBA_LOCKS synonym for DBA_LOCK view, *5-104*
DBA_LOG_GROUP_COLUMNS view, *5-104*
DBA_LOG_GROUPS view, *5-105*
DBA_LOGMNR_LOG view, *5-105*
DBA_LOGMNR_PURGED_LOG view, *5-106*
DBA_LOGMNR_SESSION view, *5-106*
DBA_LOGSTDBY_EDS_SUPPORTED view, *5-107*
DBA_LOGSTDBY_EDS_TABLES view, *5-107*
DBA_LOGSTDBY_EVENTS view, *5-107*
DBA_LOGSTDBY_HISTORY view, *5-108*
DBA_LOGSTDBY_LOG view, *5-110*
DBA_LOGSTDBY_NOT_UNIQUE view, *5-111*
DBA_LOGSTDBY_PARAMETERS view, *5-112*
DBA_LOGSTDBY_PLSQL_MAP view, *5-113*
DBA_LOGSTDBY_PLSQL_SUPPORT view, *5-114*
DBA_LOGSTDBY_PROGRESS view, *5-114*
DBA_LOGSTDBY_SKIP view, *5-114*
DBA_LOGSTDBY_SKIP_TRANSACTION view, *5-115*
DBA_LOGSTDBY_UNSUPPORTED view, *5-115*
DBA_LOGSTDBY_UNSUPPORTED_TABLE view
  LOGSTDBY_UNSUPPORTED_TABLES synonym, *6-104*
DBA_MEASURE_FOLDER_CONTENTS view, *5-117*
DBA_MEASURE_FOLDER_SUBFOLDERS view, *5-117*
DBA_MEASURE_FOLDERS view, *5-117*
DBA_METADATA_PROPERTIES view, *5-117*
DBA_METHOD_PARAMS view, *5-117*
DBA_METHOD_RESULTS view, *5-118*
DBA_MINING_MODEL_ATTRIBUTES view, *5-118*
DBA_MINING_MODEL_SETTINGS view, *5-118*
DBA_MINING_MODEL_TABLES view, *5-118*
DBA_MINING_MODELS view, *5-119*
DBA_MVIEW_AGGREGATES view, *5-119*
DBA_MVIEW_ANALYSIS view, *5-119*
DBA_MVIEW_COMMENTS view, *5-119*
DBA_MVIEW_DETAIL_PARTITION view, *5-119*
DBA_MVIEW_DETAIL_RELATIONS view, *5-119*
DBA_MVIEW_DETAIL_SUBPARTITION view, *5-120*
DBA_MVIEW_JOINS view, *5-120*
DBA_MVIEW_KEYS view, *5-120*
DBA_MVIEW_LOG_FILTER_COLS view, *5-120*
DBA_MVIEW_LOGS view, *5-120*
DBA_MVIEW_REFRESH_TIMES view, *5-120*
DBA_MVIEWS view, *5-121*
DBA_NESTED_TABLE_COLS view, *5-121*
DBA_NESTED_TABLES view, *5-121*
DBA_NETWORK_ACL_PRIVILEGES view, *5-121*
DBA_NETWORK_ACLS view, *5-122*
DBA_OBJ_AUDIT_OPTS view, *5-122*
DBA_OBJ_COLATTRS view, *5-124*
DBA_OBJECT_SIZE view, *5-124*
DBA_OBJECT_TABLES view, *5-125*
DBA_OBJECT_USAGE view, *5-125*
DBA_OBJECTS view, *5-126*
DBA_OBJECTS_AE view, *5-126*
DBA_OPANCILLARY view, *5-126*
DBA_OPARGUMENTS view, *5-126*
DBA_OPBINDINGS view, *5-126*
DBA_OPERATOR_COMMENTS view, *5-127*
DBA_OPERATORS view, *5-127*
DBA_OPTSTAT_OPERATION_TASKS view, *5-127*

DBA_OPTSTAT_OPERATIONS view, *5-128*
DBA_ORPHAN_KEY_TABLE view, *5-129*
DBA_OUTLINE_HINTS view, *5-130*
DBA_OUTLINES view, *5-131*
DBA_OUTSTANDING_ALERTS view, *5-132*
DBA_PARALLEL_EXECUTE_CHUNKS view, *5-133*
DBA_PARALLEL_EXECUTE_TASKS view, *5-134*
DBA_PART_COL_STATISTICS view, *5-135*
DBA_PART_HISTOGRAMS view, *5-135*
DBA_PART_INDEXES view, *5-136*
DBA_PART_KEY_COLUMNS view, *5-136*
DBA_PART_LOBS view, *5-136*
DBA_PART_TABLES view, *5-136*
DBA_PARTIAL_DROP_TABS view, *5-137*
DBA_PDB_HISTORY view, *5-137*
DBA_PDB_SAVED_STATES view, *5-138*
DBA_PDBS view, *5-138*
DBA_PENDING_CONV_TABLES view, *5-140*
DBA_PENDING_TRANSACTIONS view, *5-140*
DBA_PLSQL_COLL_TYPES view, *5-140*
DBA_PLSQL_OBJECT_SETTINGS view, *5-140*
DBA_PLSQL_TYPE_ATTRS view, *5-140*
DBA_PLSQL_TYPES view, *5-141*
DBA_POLICIES view, *5-141*
DBA_POLICY_ATTRIBUTES view, *5-141*
DBA_POLICY_CONTEXTS view, *5-141*
DBA_POLICY_GROUPS view, *5-141*
DBA_PRIV_AUDIT_OPTS view, *5-142*
DBA_PRIV_CAPTURES view, *5-142*
DBA_PROCEDURES view, *5-143*
DBA_PROFILES view, *5-143*
DBA_PROPAGATION view, *5-144*
DBA_PROXIES view, *5-144*
DBA_QUEUE_SCHEDULES view, *5-144*
DBA_QUEUE_SUBSCRIBERS view, *5-145*
DBA_QUEUE_TABLES view, *5-145*
DBA_QUEUES view, *5-145*
DBA_RCHILD view, *5-145*
DBA_RECOVERABLE_SCRIPT view, *5-145*
DBA_RECOVERABLE_SCRIPT_BLOCKS view, *5-146*
DBA_RECOVERABLE_SCRIPT_ERRORS view, *5-146*
DBA_RECOVERABLE_SCRIPT_HIST view, *5-147*
DBA_RECOVERABLE_SCRIPT_PARAMS view, *5-147*
DBA_RECYCLEBIN view, *5-148*
DBA_REDEFINITION_ERRORS view, *5-149*
DBA_REDEFINITION_OBJECTS view, *5-150*
DBA_REDEFINITION_STATUS view, *5-150*
DBA_REFRESH view, *5-151*
DBA_REFRESH_CHILDREN view, *5-151*
DBA_REFS view, *5-151*
DBA_REGISTERED_ARCHIVED_LOG view, *5-151*
DBA_REGISTERED_MVIEW_GROUPS view, *5-152*
DBA_REGISTERED_MVIEWS view, *5-153*
DBA_REGISTRY view, *5-153*
DBA_REGISTRY_HIERARCHY view, *5-154*
DBA_REGISTRY_HISTORY view, *5-154*
DBA_REGISTRY_LOG view, *5-155*
DBA_REGISTRY_SQLPATCH view, *5-155*
DBA_REPAIR_TABLE view, *5-156*
DBA_REPCAT view, *2-3*
DBA_REPCAT_EXCEPTIONS view, *2-3*
DBA_REPCAT_TEMPLATE_OBJECTS view, *2-3*
DBA_REPCAT_TEMPLATE_PARMS view, *2-3*
DBA_REPCAT_TEMPLATE_SITES view, *2-3*
DBA_REPCAT_USER_AUTHORIZATIONS view, *2-3*
DBA_REPCAT_USER_PARM_VALUES view, *2-3*
DBA_REPCATLOG view, *2-3*
DBA_REPCOLUMN view, *2-4*
DBA_REPCOLUMN_GROUP view, *2-4*
DBA_REPCONFLICT view, *2-4*
DBA_REPDDL view, *2-4*
DBA_REPEXTENSIONS view, *2-4*
DBA_REPFLAVOR_COLUMNS view, *2-4*
DBA_REPFLAVOR_OBJECTS view, *2-4*
DBA_REPFLAVORS view, *2-4*
DBA_REPGENERATED view, *2-4*
DBA_REPGENOBJECTS view, *2-4*
DBA_REPGROUP view, *2-4*
DBA_REPGROUP_PRIVILEGES view, *2-4*
DBA_REPGROUPED_COLUMN view, *2-4*
DBA_REPKEY_COLUMNS view, *2-4*
DBA_REPL_DBNAME_MAPPING view, *5-157*
DBA_REPOBJECT view, *2-4*
DBA_REPPARAMETER_COLUMN view, *2-4*
DBA_REPPRIORITY view, *2-4*
DBA_REPPRIORITY_GROUP view, *2-4*
DBA_REPPROP view, *2-4*
DBA_REPRESOL_STATS_CONTROL view, *2-4*
DBA_REPRESOLUTION view, *2-4*
DBA_REPRESOLUTION_METHOD view, *2-4*
DBA_REPRESOLUTION_STATISTICS view, *2-4*
DBA_REPSCHEMA view, *2-4*
DBA_REPSITES view, *2-4*
DBA_REPSITES_NEW view, *2-4*
DBA_RESOURCE_INCARNATIONS view, *5-157*
DBA_RESUMABLE view, *5-158*
DBA_REWRITE_EQUIVALENCES view, *5-159*
DBA_RGROUP view, *5-159*
DBA_ROLE_PRIVS view, *5-160*
DBA_ROLES view, *5-161*
DBA_ROLLBACK_SEGS view, *5-161*
DBA_ROLLING_DATABASES view, *5-162*
DBA_ROLLING_EVENTS view, *5-163*
DBA_ROLLING_PARAMETERS view, *5-164*
DBA_ROLLING_PLAN view, *5-164*
DBA_ROLLING_STATISTICS view, *5-165*
DBA_ROLLING_STATUS view, *5-165*
DBA_ROLLING_UNSUPPORTED view, *5-166*
DBA_RSRC_CATEGORIES view, *5-167*
DBA_RSRC_CONSUMER_GROUP_PRIVS view, *5-168*

DBA_RSRC_CONSUMER_GROUPS view, *5-168*
DBA_RSRC_GROUP_MAPPINGS view, *5-169*
DBA_RSRC_IO_CALIBRATE view, *5-169*
DBA_RSRC_MANAGER_SYSTEM_PRIVS view, *5-170*
DBA_RSRC_MAPPING_PRIORITY view, *5-170*
DBA_RSRC_PLAN_DIRECTIVES view, *5-170*
DBA_RSRC_PLANS view, *5-173*
DBA_RULE_SET_RULES view, *5-174*
DBA_RULE_SETS view, *5-174*
DBA_RULES view, *5-175*
DBA_SCHEDULER_CHAIN_RULES view, *5-175*
DBA_SCHEDULER_CHAIN_STEPS view, *5-175*
DBA_SCHEDULER_CHAINS view, *5-175*
DBA_SCHEDULER_CREDENTIALS view, *5-175*
DBA_SCHEDULER_DB_DESTS view, *5-176*
DBA_SCHEDULER_DESTS view, *5-176*
DBA_SCHEDULER_EXTERNAL_DESTS view, *5-176*
DBA_SCHEDULER_FILE_WATCHERS view, *5-176*
DBA_SCHEDULER_GLOBAL_ATTRIBUTE view, *5-177*
DBA_SCHEDULER_GROUP_MEMBERS view, *5-177*
DBA_SCHEDULER_GROUPS view, *5-177*
DBA_SCHEDULER_JOB_ARGS view, *5-177*
DBA_SCHEDULER_JOB_CLASSES view, *5-177*
DBA_SCHEDULER_JOB_DESTS view, *5-178*
DBA_SCHEDULER_JOB_LOG view, *5-178*
DBA_SCHEDULER_JOB_ROLES view, *5-178*
DBA_SCHEDULER_JOB_RUN_DETAILS view, *5-180*
DBA_SCHEDULER_JOBS view, *5-180*
DBA_SCHEDULER_NOTIFICATIONS view, *5-180*
DBA_SCHEDULER_PROGRAM_ARGS view, *5-180*
DBA_SCHEDULER_PROGRAMS view, *5-180*
DBA_SCHEDULER_REMOTE_DATABASES view, *5-181*
DBA_SCHEDULER_REMOTE_JOBSTATE view, *5-181*
DBA_SCHEDULER_RUNNING_CHAINS view, *5-181*
DBA_SCHEDULER_RUNNING_JOBS view, *5-181*
DBA_SCHEDULER_SCHEDULES view, *5-181*
DBA_SCHEDULER_WINDOW_DETAILS view, *5-182*
DBA_SCHEDULER_WINDOW_GROUPS view, *5-182*
DBA_SCHEDULER_WINDOW_LOG view, *5-182*
DBA_SCHEDULER_WINDOWS view, *5-182*
DBA_SCHEDULER_WINGROUP_MEMBERS view, *5-182*
DBA_SEC_RELEVANT_COLS view, *5-183*
DBA_SECONDARY_OBJECTS view, *5-183*
DBA_SEGMENTS view, *5-183*
DBA_SEGMENTS_OLD view, *5-186*
DBA_SENSITIVE_COLUMN_TYPES view, *5-187*
DBA_SENSITIVE_DATA view, *5-187*
DBA_SEQUENCES view, *5-188*
DBA_SERVER_REGISTRY view, *5-188*
DBA_SERVICES view, *5-188*
DBA_SOURCE view, *5-189*
DBA_SOURCE_AE view, *5-189*
DBA_SQL_MANAGEMENT_CONFIG view, *5-189*
DBA_SQL_PATCHES view, *5-189*
DBA_SQL_PLAN_BASELINES view, *5-190*
DBA_SQL_PLAN_DIR_OBJECTS view, *5-192*
DBA_SQL_PLAN_DIRECTIVES view, *5-193*
DBA_SQL_PROFILES view, *5-193*
DBA_SQL_TRANSLATION_PROFILES view, *5-195*
DBA_SQL_TRANSLATIONS view, *5-195*
DBA_SQLJ_TYPE_ATTRS view, *5-195*
DBA_SQLJ_TYPE_METHODS view, *5-195*
DBA_SQLJ_TYPES view, *5-195*
DBA_SQLSET view, *5-196*
DBA_SQLSET_BINDS view, *5-196*
DBA_SQLSET_PLANS view, *5-196*
DBA_SQLSET_REFERENCES view, *5-196*
DBA_SQLSET_STATEMENTS view, *5-196*
DBA_SQLTUNE_BINDS view, *5-198*
DBA_SQLTUNE_PLANS view, *5-198*
DBA_SQLTUNE_RATIONALE_PLAN view, *5-201*
DBA_SQLTUNE_STATISTICS view, *5-202*
DBA_SR_GRP_STATUS view, *5-203*
DBA_SR_GRP_STATUS_ALL view, *5-204*
DBA_SR_OBJ view, *5-206*
DBA_SR_OBJ_ALL view, *5-207*
DBA_SR_OBJ_STATUS view, *5-207*
DBA_SR_OBJ_STATUS_ALL view, *5-208*
DBA_SR_PARTN_OPS view, *5-209*
DBA_SR_STLOG_EXCEPTIONS view, *5-210*
DBA_SR_STLOG_STATS view, *5-211*
DBA_SSCR_CAPTURE view, *5-212*
DBA_SSCR_RESTORE view, *5-212*
DBA_STAT_EXTENSIONS view, *5-213*
DBA_STMT_AUDIT_OPTS view, *5-213*
DBA_STORED_SETTINGS view, *5-214*
DBA_STREAMS_ADD_COLUMN view, *6-1*
DBA_STREAMS_ADMINISTRATOR view, *6-1*
DBA_STREAMS_COLUMNS view, *6-2*
DBA_STREAMS_DELETE_COLUMN view, *6-2*
DBA_STREAMS_GLOBAL_RULES view, *6-3*
DBA_STREAMS_KEEP_COLUMNS view, *6-3*
DBA_STREAMS_MESSAGE_CONSUMERS view, *6-3*
DBA_STREAMS_MESSAGE_RULES view, *6-4*
DBA_STREAMS_NEWLY_SUPPORTED view, *6-4*
DBA_STREAMS_RENAME_COLUMN view, *6-4*
DBA_STREAMS_RENAME_SCHEMA view, *6-5*
DBA_STREAMS_RENAME_TABLE view, *6-5*
DBA_STREAMS_RULES view, *6-5*
DBA_STREAMS_SCHEMA_RULES view, *6-6*
DBA_STREAMS_SPLIT_MERGE view, *6-6*
DBA_STREAMS_SPLIT_MERGE_HIST view, *6-8*
DBA_STREAMS_STMT_HANDLERS view, *6-10*
DBA_STREAMS_STMTS view, *6-10*
DBA_STREAMS_TABLE_RULES view, *6-10*
DBA_STREAMS_TP_COMPONENT view, *6-10*
DBA_STREAMS_TP_COMPONENT_LINK view, *6-11*
DBA_STREAMS_TP_COMPONENT_STAT view, *6-12*

DBA_STREAMS_TP_DATABASE view, *6-12*
DBA_STREAMS_TP_PATH_BOTTLENECK view, *6-13*
DBA_STREAMS_TP_PATH_STAT view, *6-14*
DBA_STREAMS_TRANSFORM_FUNCTION view, *6-14*
DBA_STREAMS_TRANSFORMATIONS view, *6-14*
DBA_STREAMS_UNSUPPORTED view, *6-15*
DBA_SUBPART_COL_STATISTICS view, *6-16*
DBA_SUBPART_HISTOGRAMS view, *6-16*
DBA_SUBPART_KEY_COLUMNS view, *6-16*
DBA_SUBPARTITION_TEMPLATES view, *6-16*
DBA_SUBSCR_REGISTRATIONS view, *6-16*
DBA_SUPPLEMENTAL_LOGGING view, *6-18*
DBA_SYNC_CAPTURE view, *6-18*
DBA_SYNC_CAPTURE_PREPARED_TABS view, *6-19*
DBA_SYNC_CAPTURE_TABLES view, *6-19*
DBA_SYNONYMS view, *6-19*
DBA_SYS_PRIVS view, *6-19*
DBA_TAB_COL_STATISTICS view, *6-20*
DBA_TAB_COLS view, *6-20*
DBA_TAB_COLUMNS view, *6-23*
DBA_TAB_COMMENTS view, *6-26*
DBA_TAB_HISTGRM_PENDING_STATS view, *6-26*
DBA_TAB_HISTOGRAMS view
  DBA_HISTOGRAMS synonym, *5-86*
DBA_TAB_IDENTITY_COLS view, *6-26*
DBA_TAB_MODIFICATIONS view, *6-26*
DBA_TAB_PARTITIONS view, *6-27*
DBA_TAB_PENDING_STATS view, *6-27*
DBA_TAB_PRIVS view, *6-27*
DBA_TAB_STAT_PREFS view, *6-28*
DBA_TAB_STATISTICS view, *6-28*
DBA_TAB_STATS_HISTORY view, *6-28*
DBA_TAB_SUBPARTITIONS view, *6-28*
DBA_TABLES view, *6-28*
DBA_TABLESPACE_GROUPS view, *6-29*
DBA_TABLESPACE_THRESHOLDS view, *6-29*
DBA_TABLESPACE_USAGE_METRICS view, *6-30*
DBA_TABLESPACES view, *6-30*
DBA_TEMP_FILES view, *6-33*
DBA_TEMP_FREE_SPACE view, *6-33*
DBA_THRESHOLDS view, *6-34*
DBA_TRANSFORMATIONS view, *6-35*
DBA_TRIGGER_COLS view, *6-35*
DBA_TRIGGER_ORDERING view, *6-35*
DBA_TRIGGERS view, *6-35*
DBA_TS_QUOTAS view, *6-36*
DBA_TSDP_IMPORT_ERRORS view, *6-36*
DBA_TSDP_POLICY_CONDITION view, *6-37*
DBA_TSDP_POLICY_FEATURE view, *6-37*
DBA_TSDP_POLICY_PARAMETER view, *6-38*
DBA_TSDP_POLICY_PROTECTION view, *6-38*
DBA_TSDP_POLICY_TYPE view, *6-39*
DBA_TSM_DESTINATION view, *6-39*
DBA_TSM_SOURCE view, *6-40*
DBA_TSTZ_TAB_COLS view, *6-41*
DBA_TSTZ_TABLES view, *6-41*
DBA_TUNE_MVIEW view, *6-41*
DBA_TYPE_ATTRS view, *6-42*
DBA_TYPE_METHODS view, *6-42*
DBA_TYPE_VERSIONS view, *6-42*
DBA_TYPES view, *6-42*
DBA_UNDO_EXTENTS view, *6-42*
DBA_UNUSED_COL_TABS view, *6-43*
DBA_UNUSED_OBJPRIVS view, *6-44*
DBA_UNUSED_OBJPRIVS_PATH view, *6-44*
DBA_UNUSED_PRIVS view, *6-45*
DBA_UNUSED_SYSPRIVS view, *6-46*
DBA_UNUSED_SYSPRIVS_PATH view, *6-47*
DBA_UNUSED_USERPRIVS view, *6-48*
DBA_UNUSED_USERPRIVS_PATH view, *6-49*
DBA_UPDATABLE_COLUMNS view, *6-49*
DBA_USED_OBJPRIVS view, *6-50*
DBA_USED_OBJPRIVS_PATH view, *6-51*
DBA_USED_PRIVS view, *6-52*
DBA_USED_PUBPRIVS view, *6-53*
DBA_USED_SYSPRIVS view, *6-54*
DBA_USED_SYSPRIVS_PATH view, *6-55*
DBA_USED_USERPRIVS view, *6-55*
DBA_USED_USERPRIVS_PATH view, *6-56*
DBA_USERS view, *6-57*
DBA_USERS_WITH_DEFPWD view, *6-59*
DBA_USTATS view, *6-59*
DBA_VARRAYS view, *6-60*
DBA_VIEWS view, *6-60*
DBA_VIEWS_AE view, *6-60*
DBA_WAITERS view, *6-60*
DBA_WALLET_ACES view, *6-61*
DBA_WALLET_ACLS view, *6-61*
DBA_WARNING_SETTINGS view, *6-62*
DBA_WI_CAPTURE_FILES view, *6-62*
DBA_WI_JOBS view, *6-62*
DBA_WI_OBJECTS view, *6-63*
DBA_WI_PATTERN_ITEMS view, *6-63*
DBA_WI_PATTERNS view, *6-63*
DBA_WI_STATEMENTS view, *6-64*
DBA_WI_TEMPLATE_EXECUTIONS view, *6-64*
DBA_WI_TEMPLATES view, *6-65*
DBA_WM_SYS_PRIVS view, *2-5*
DBA_WM_VT_ERRORS view, *2-5*
DBA_WORKLOAD_ACTIVE_USER_MAP view, *6-65*
DBA_WORKLOAD_CAPTURES view, *6-66*
DBA_WORKLOAD_CONNECTION_MAP view, *6-68*
DBA_WORKLOAD_FILTERS view, *6-69*
DBA_WORKLOAD_REPLAY_DIVERGENCE view, *6-69*
DBA_WORKLOAD_REPLAY_SCHEDULES view, *6-71*
DBA_WORKLOAD_REPLAYS view, *6-71*
DBA_WORKLOAD_SCHEDULE_CAPTURES view, *6-75*

DBA_WORKLOAD_SCHEDULE_ORDERING view, 6-76
DBA_WORKLOAD_USER_MAP view, 6-77
DBA_WORKSPACE_SESSIONS view, 2-5
DBA_WORKSPACES view, 2-6
DBA_XML_INDEXES view, 6-77
DBA_XML_NESTED_TABLES view, 6-78
DBA_XML_OUT_OF_LINE_TABLES view, 6-78
DBA_XML_SCHEMA_ATTRIBUTES view, 6-78
DBA_XML_SCHEMA_COMPLEX_TYPES view, 6-78
DBA_XML_SCHEMA_ELEMENTS view, 6-78
DBA_XML_SCHEMA_NAMESPACES view, 6-79
DBA_XML_SCHEMA_SIMPLE_TYPES view, 6-79
DBA_XML_SCHEMA_SUBSTGRP_HEAD view, 6-79
DBA_XML_SCHEMA_SUBSTGRP_MBRS view, 6-79
DBA_XML_SCHEMAS view, 6-79
DBA_XML_TAB_COLS view, 6-80
DBA_XML_TABLES view, 6-80
DBA_XML_VIEW_COLS view, 6-80
DBA_XML_VIEWS view, 6-80
DBA_XS_AUDIT_POLICY_OPTIONS view, 6-80
DBA_XS_AUDIT_TRAIL view, 6-81
DBA_XS_ENB_AUDIT_POLICIES view, 6-82
DBA_XSTREAM_ADMINISTRATOR view, 6-83
DBA_XSTREAM_INBOUND view, 6-83
DBA_XSTREAM_INBOUND_PROGRESS view, 6-83
DBA_XSTREAM_OUT_SUPPORT_MODE view, 6-83
DBA_XSTREAM_OUTBOUND view, 6-84
DBA_XSTREAM_OUTBOUND_PROGRESS view, 6-84
DBA_XSTREAM_RULES view, 6-84
DBA_XSTREAM_SPLIT_MERGE view, 6-84
DBA_XSTREAM_SPLIT_MERGE_HIST view, 6-86
DBA_XSTREAM_STMT_HANDLERS view, 6-88
DBA_XSTREAM_STMTS view, 6-88
DBA_XSTREAM_TRANSFORMATIONS view, 6-89
DBA_ZONEMAP_MEASURES view, 6-89
DBA_ZONEMAPS view, 6-89
DBFIPS_140 initialization parameter, 1-76
DBFS_CONTENT view, 6-90
DBFS_CONTENT_PROPERTIES view, 6-92
DBMS_ALERT_INFO view, 6-92
DBMS_LOCK_ALLOCATED view, 6-92
DBMS_STATS package, 2-119, 2-173, 3-111, 3-114, 3-137, 5-95, 5-121, 6-20, 6-23, 6-28, 6-177, 6-202, 6-205
DBMSPOOL.SQL script, B-3
DBWR_IO_SLAVES initialization parameter, 1-77
DDL_LOCK_TIMEOUT initialization parameter, 1-78
DEFCALL view, 2-4
DEFCALLDEST view, 2-4
DEFDEFAULTDEST view, 2-4
DEFERRCOUNT view, 2-4
DEFERRED_SEGMENT_CREATION initialization parameter, 1-78
DEFERROR view, 2-4
DEFLOB view, 2-5
DEFPROPAGATOR view, 2-5
DEFSCHEDULE view, 2-5
DEFTRAN view, 2-5
DEFTRANDEST view, 2-5
DEPTREE view, 6-93
destination
  archiving redo log files, 1-124
  overriding default, 1-125
  USER_DUMP_DEST, 1-253
DG_BROKER_CONFIG_FILEn initialization parameter, 1-79
DG_BROKER_START initialization parameter, 1-80
DIAGNOSTIC_DEST initialization parameter, 1-80
DICT synonym for DICTIONARY, 6-93
DICT_COLUMNS view, 6-93
DICTIONARY view
  DICT synonym, 6-93
disk drives
  archiving destination, 1-124
DISK_ASYNCH_IO initialization parameter, 1-81
dispatcher processes
  maximum number, 1-141
DISPATCHERS initialization parameter, 1-82
distinguished name, 1-202
DISTRIBUTED_LOCK_TIMEOUT initialization parameter, 1-84
DM_USER_MODELS view, 6-94
DML_LOCKS initialization parameter, 1-84
DNFS_BATCH_SIZE initialization parameter, 1-85
DOCUMENT_LINKS view, 6-95
DST_UPGRADE_INSERT_CONV initialization parameter, 1-86
dump files, 1-142
dynamic performance tables
  CATALOG.SQL script, 7-1
  public synonyms (V$), 7-2
  views (V_$), 7-2

## E

embedded initialization parameter files, 1-100
ENABLE_DDL_LOGGING initialization parameter, 1-87
ENABLE_GOLDENGATE_REPLICATION initialization parameter, 1-88
ENABLE_PLUGGABLE_DATABASE initialization parameter, 1-89
enqueues
  AJV snapshot refresh, D-2
  ALTER SYSTEM SET PARAMETER = VALUE, D-2
  backup/restore, D-1
  being written redo log, D-3
  bind, D-1
  buffer cache management, D-1
  controlfile transaction, D-1

enqueues *(continued)*
cross-instance call invocation, *D-1*
database mount, *D-1*
datafile, *D-1*
direct loader index creation, *D-1*
disabling, *1-85*
distributed recovery process, *D-1*
distributed transaction, *D-1*
DML, *D-3*
extend table, *D-3*
file set, *D-1*
high-water lock, *D-2*
instance attribute lock, *D-3*
instance number, *D-2*
instance recovery, *D-2*
instance registration lock, *D-3*
instance state, *D-2*
job queue, *D-2*
library cache invalidation, *D-2*
library cache lock, *D-2*
library cache pin, *D-2*
log start or switch, *D-2*
media recovery, *D-2*
mount definition, *D-2*
object reuse, *D-2*
parallel slave synchronization, *D-2*
parallel slaves, *D-2*
password file, *D-2*
process startup, *D-2*
redo log "kick", *D-2*
redo thread, *D-2*
row cache, *D-2*
row wait, *D-2*
sequence number, *D-2*
sequence number value, *D-2*
SMON, *D-2*
sort segment, *D-2*
space management transaction, *D-2*
synchronized replication, *D-2*
system commit number, *D-2*
temporary segment, *D-3*
temporary table, *D-3*
temporary table object, *D-3*
thread checkpoint, *D-3*
transaction, *D-3*
transaction recovery, *D-3*
undo segment, serialization, *D-3*
user name, *D-3*
user-defined locks, *D-3*
enterprise roles, *1-202*
ERROR_SIZE view, *6-95*
EVENT initialization parameter, *1-90*
EXAFUSION_ENABLED initialization parameter, *1-90*
EXCEPTIONS table, *6-96*
extent
allocating to instance, *1-111*
extent *(continued)*

## F

FAL_CLIENT initialization parameter, *1-91*
FAL_SERVER initialization parameter, *1-91*
FAST_START_MTTR_TARGET initialization parameter, *1-92*
FAST_START_PARALLEL_ROLLBACK initialization parameter, *1-92*
FILE_MAPPING initialization parameter, *1-93*
FILEIO_NETWORK_ADAPTERS initialization parameter, *1-94*
filenames
case significance, *1-4*
FILESYSTEMIO_OPTIONS initialization parameter, *1-94*
FIXED_DATE initialization parameter, *1-95*
FLASHBACK_TRANSACTION_QUERY view, *6-96*
free space list
locating space, *1-111*
FREELIST GROUPS clause, *1-111*

## G

GCS_SERVER_PROCESSES initialization parameter, *1-95*
GLOBAL_CONTEXT view, *6-97*
GLOBAL_CONTEXT_POOL_SIZE initialization parameter, *1-96*
GLOBAL_NAME view, *6-97*
GLOBAL_NAMES initialization parameter, *1-96*
GLOBAL_TXN_PROCESSES initialization parameter, *1-97*
GV$ views
for Real Application Clusters, *7-2*

## H

HASH_AREA_SIZE initialization parameter, *1-97*
HEAT_MAP initialization parameter, *1-98*
Heterogeneous Services
agents, *1-99*
HI_SHARED_MEMORY_ADDRESS initialization parameter, *1-99*
hints
precedence over initialization parameter settings, *1-230*
HS_ALL_CAPS view, *6-97*
HS_ALL_DD view, *6-98*
HS_ALL_INITS view, *6-98*
HS_AUTOREGISTER initialization parameter, *1-99*
HS_BASE_CAPS view, *6-98*
HS_BASE_DD view, *6-99*
HS_CLASS_CAPS view, *6-99*
HS_CLASS_DD view, *6-99*

HS_CLASS_INIT view, *6-100*
HS_FDS_CLASS view, *6-100*
HS_FDS_INST view, *6-100*
HS_INST_CAPS view, *6-101*
HS_INST_DD view, *6-101*
HS_INST_INIT view, *6-102*

## I

I/O
  reading multiple blocks, *1-63*
  redo log files, *1-137*
IDEPTREE view, *6-102*
IFILE initialization parameter, *1-5*, *1-99*
IM column store. See In-Memory Column Store, *1-100*
IN-list iterators
  optimizer use of, *1-171*
In-Memory Column Store, *1-100*, *1-105*, *1-107*, *1-108*
IND synonym for USER_INDEXES view, *6-102*
INDEX_HISTOGRAM view, *6-102*
INDEX_STATS view, *6-103*
INIT.ORA file *See* initialization parameter file
  *See* initialization parameter file
initialization parameter file
  common file, *1-100*
  embedded, *1-100*
  INITDW.ORA, *1-4*
  line continuation character, *1-5*
  overview, *1-3*
initialization parameters
  ACTIVE_INSTANCE_COUNT, *1-23*
  altering, *1-7*
  AQ_TM_PROCESSES, *1-23*
  ARCHIVE_LAG_TARGET, *1-24*
  ASM_DISKGROUPS, *1-25*
  ASM_DISKSTRING, *1-25*
  ASM_POWER_LIMIT, *1-26*
  ASM_PREFERRED_READ_FAILURE_GROUPS, *1-27*
  AUDIT_FILE_DEST, *1-27*
  AUDIT_SYS_OPERATIONS, *1-28*
  AUDIT_SYSLOG_LEVEL, *1-29*
  AUDIT_TRAIL, *1-30*
  AWR_SNAPSHOT_TIME_OFFSET, *1-33*
  BACKGROUND_CORE_DUMP, *1-33*
  BACKGROUND_DUMP_DEST, *1-34*
  BACKUP_TAPE_IO_SLAVES, *1-35*
  basic, *1-3*
  BITMAP_MERGE_AREA_SIZE, *1-35*
  BLANK_TRIMMING, *1-36*
  case significance in filenames, *1-4*
  CDB_COMPATIBLE, *1-161*
  CIRCUITS, *1-37*
  CLIENT_RESULT_CACHE_LAG, *1-37*
  CLIENT_RESULT_CACHE_SIZE, *1-38*
  CLONEDB, *1-38*
initialization parameters *(continued)*
  CLUSTER_DATABASE, *1-39*
  CLUSTER_DATABASE_INSTANCES, *1-39*
  CLUSTER_INTERCONNECTS, *1-40*
  COMMIT_LOGGING, *1-41*
  COMMIT_POINT_STRENGTH, *1-41*
  COMMIT_WAIT, *1-42*
  COMMIT_WRITE, *1-42*
  COMMON_USER_PREFIX, *1-43*
  COMPATIBLE, *1-44*
  CONNECTION_BROKERS, *1-45*
  CONTROL_FILE_RECORD_KEEP_TIME, *1-46*
  CONTROL_FILES, *1-47*
  CONTROL_MANAGEMENT_PACK_ACCESS, *1-48*
  CORE_DUMP_DEST, *1-48*
  CPU_COUNT, *1-49*
  CREATE_BITMAP_AREA_SIZE, *1-50*
  CREATE_STORED_OUTLINES, *1-50*
  CURSOR_BIND_CAPTURE_DESTINATION, *1-51*
  CURSOR_SHARING, *1-52*
  CURSOR_SPACE_FOR_TIME, *1-53*
  DB_BIG_TABLE_CACHE_PERCENT_TARGET, *1-54*
  DB_BLOCK_BUFFERS, *1-56*
  DB_BLOCK_CHECKING, *1-57*
  DB_BLOCK_CHECKSUM, *1-58*
  DB_BLOCK_SIZE, *1-59*
  DB_CACHE_ADVICE, *1-59*
  DB_CACHE_SIZE, *1-60*
  DB_CREATE_FILE_DEST, *1-61*
  DB_CREATE_ONLINE_LOG_DEST_n, *1-62*
  DB_DOMAIN, *1-62*
  DB_FILE_MULTIBLOCK_READ_COUNT, *1-63*
  DB_FILE_NAME_CONVERT, *1-64*
  DB_FILES, *1-65*
  DB_FLASH_CACHE_FILE, *1-66*
  DB_FLASH_CACHE_SIZE, *1-66*
  DB_FLASHBACK_RETENTION_TARGET, *1-67*
  DB_INDEX_COMPRESSION_INHERITANCE, *1-68*
  DB_KEEP_CACHE_SIZE, *1-68*
  DB_LOST_WRITE_PROTECT, *1-69*
  DB_NAME, *1-70*
  DB_nK_CACHE_SIZE, *1-54*
  DB_RECOVERY_FILE_DEST, *1-70*
  DB_RECOVERY_FILE_DEST_SIZE, *1-71*
  DB_RECYCLE_CACHE_SIZE, *1-72*
  DB_SECUREFILE, *1-72*
  DB_ULTRA_SAFE, *1-74*
  DB_UNIQUE_NAME, *1-74*
  DB_UNRECOVERABLE_SCN_TRACKING, *1-75*
  DB_WRITER_PROCESSES, *1-76*
  DBFIPS_140, *1-76*
  DBWR_IO_SLAVES, *1-77*

initialization parameters *(continued)*
DDL_LOCK_TIMEOUT, *1-78*
DEFERRED_SEGMENT_CREATION, *1-78*
derived, *1-2*
DG_BROKER_CONFIG_FILEn, *1-79*
DG_BROKER_START, *1-80*
DIAGNOSTIC_DEST, *1-80*
DISK_ASYNCH_IO, *1-81*
DISPATCHERS, *1-82*
display current settings, *1-21*
DISTRIBUTED_LOCK_TIMEOUT, *1-84*
DML_LOCKS, *1-84*
DNFS_BATCH_SIZE, *1-85*
DST_UPGRADE_INSERT_CONV, *1-86*
embedded initialization parameter files, *1-5*
ENABLE_DDL_LOGGING, *1-87*
ENABLE_GOLDENGATE_REPLICATION, *1-88*
ENABLE_PLUGGABLE_DATABASE, *1-89*
EVENT, *1-90*
EXAFUSION_ENABLED, *1-90*
FAL_CLIENT, *1-91*
FAL_SERVER, *1-91*
FAST_START_MTTR_TARGET, *1-92*
FAST_START_PARALLEL_ROLLBACK, *1-92*
FILE_MAPPING, *1-93*
FILEIO_NETWORK_ADAPTERS, *1-94*
FILESYSTEMIO_OPTIONS, *1-94*
FIXED_DATE, *1-95*
functional category, *1-7*
GCS_SERVER_PROCESSES, *1-95*
GLOBAL_CONTEXT_POOL_SIZE, *1-96*
GLOBAL_NAMES, *1-96*
GLOBAL_TXN_PROCESSES, *1-97*
HASH_AREA_SIZE, *1-97*
HEAT_MAP, *1-98*
HI_SHARED_MEMORY_ADDRESS, *1-99*
HS_AUTOREGISTER, *1-99*
IFILE, *1-5*, *1-99*
INMEMORY_CLAUSE_DEFAULT, *1-100*
INMEMORY_FORCE, *1-105*
INMEMORY_MAX_POPULATE_SERVERS, *1-106*
INMEMORY_QUERY, *1-107*
INMEMORY_SIZE, *1-108*
INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT, *1-109*
INSTANCE_GROUPS, *1-110*
INSTANCE_NAME, *1-110*
INSTANCE_NUMBER, *1-111*
INSTANCE_TYPE, *1-112*
JAVA_JIT_ENABLED, *1-112*
JAVA_MAX_SESSIONSPACE_SIZE, *1-113*
JAVA_POOL_SIZE, *1-113*
JAVA_SOFT_SESSIONSPACE_LIMIT, *1-114*
JOB_QUEUE_PROCESSES, *1-115*
LARGE_POOL_SIZE, *1-115*
LDAP_DIRECTORY_ACCESS, *1-117*
initialization parameters *(continued)*
LDAP_DIRECTORY_SYSAUTH, *1-118*
LICENSE_MAX_SESSIONS, *1-118*
LICENSE_MAX_USERS, *1-119*
LICENSE_SESSIONS_WARNING, *1-120*
LISTENER_NETWORKS, *1-121*
LOCAL_LISTENER, *1-121*
LOCK_NAME_SPACE, *1-122*
LOCK_SGA, *1-122*
LOG_ARCHIVE_CONFIG, *1-123*
LOG_ARCHIVE_DEST, *1-124*
LOG_ARCHIVE_DEST_n, *1-125*
LOG_ARCHIVE_DEST_STATE_n, *1-129*
LOG_ARCHIVE_DUPLEX_DEST, *1-130*
LOG_ARCHIVE_FORMAT, *1-131*
LOG_ARCHIVE_MAX_PROCESSES, *1-133*
LOG_ARCHIVE_MIN_SUCCEED_DEST, *1-133*
LOG_ARCHIVE_TRACE, *1-134*
LOG_BUFFER, *1-137*
LOG_CHECKPOINT_INTERVAL, *1-138*
LOG_CHECKPOINT_TIMEOUT, *1-139*
LOG_CHECKPOINTS_TO_ALERT, *1-139*
LOG_FILE_NAME_CONVERT, *1-140*
MAX_DISPATCHERS, *1-141*
MAX_DUMP_FILE_SIZE, *1-141*
MAX_ENABLED_ROLES, *1-142*
MAX_SHARED_SERVERS, *1-143*
MAX_STRING_SIZE, *1-144*
MEMORY_MAX_TARGET, *1-147*
MEMORY_TARGET, *1-147*
modifiable, *1-14*
NLS_CALENDAR, *1-148*
NLS_COMP, *1-149*
NLS_CURRENCY, *1-150*
NLS_DATE_FORMAT, *1-151*
NLS_DATE_LANGUAGE, *1-152*
NLS_DUAL_CURRENCY, *1-153*
NLS_ISO_CURRENCY, *1-153*
NLS_LANGUAGE, *1-154*
NLS_LENGTH_SEMANTICS, *1-155*
NLS_NCHAR_CONV_EXCP, *1-156*
NLS_NUMERIC_CHARACTERS, *1-157*
NLS_SORT, *1-157*
NLS_TERRITORY, *1-159*
NLS_TIMESTAMP_FORMAT, *1-159*
NLS_TIMESTAMP_TZ_FORMAT, *1-160*
O7_DICTIONARY_ACCESSIBILITY, *1-162*
OBJECT_CACHE_MAX_SIZE_PERCENT, *1-163*
OBJECT_CACHE_OPTIMAL_SIZE, *1-163*
OLAP_PAGE_POOL_SIZE, *1-164*
OPEN_CURSORS, *1-164*
OPEN_LINKS, *1-165*
OPEN_LINKS_PER_INSTANCE, *1-166*
operating system-dependent values, *1-2*
OPTIMIZER_ADAPTIVE_FEATURES, *1-166*

initialization parameters *(continued)*
OPTIMIZER_ADAPTIVE_REPORTING_ONLY, *1-167*
OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES, *1-167*
OPTIMIZER_DYNAMIC_SAMPLING, *1-168*
OPTIMIZER_FEATURES_ENABLE, *1-169*
OPTIMIZER_INDEX_CACHING, *1-171*
OPTIMIZER_INDEX_COST_ADJ, *1-172*
OPTIMIZER_INMEMORY_AWARE, *1-172*
OPTIMIZER_MODE, *1-173*
OPTIMIZER_SECURE_VIEW_MERGING, *1-174*
OPTIMIZER_USE_INVISIBLE_INDEXES, *1-175*
OPTIMIZER_USE_PENDING_STATISTICS, *1-175*
OPTIMIZER_USE_SQL_PLAN_BASELINES, *1-175*
OS_AUTHENT_PREFIX, *1-176*
OS_ROLES, *1-177*
PARALLEL_ADAPTIVE_MULTI_USER, *1-177*
PARALLEL_AUTOMATIC_TUNING, *1-178*
PARALLEL_DEGREE_LIMIT, *1-179*
PARALLEL_DEGREE_POLICY, *1-180*
PARALLEL_EXECUTION_MESSAGE_SIZE, *1-181*
PARALLEL_FORCE_LOCAL, *1-182*
PARALLEL_INSTANCE_GROUP, *1-182*
PARALLEL_IO_CAP_ENABLED, *1-183*
PARALLEL_MAX_SERVERS, *1-184*
PARALLEL_MIN_PERCENT, *1-185*
PARALLEL_MIN_SERVERS, *1-186*
PARALLEL_MIN_TIME_THRESHOLD, *1-187*
PARALLEL_SERVERS_TARGET, *1-187*
PARALLEL_THREADS_PER_CPU, *1-188*
parameter files, *1-3*
PDB_FILE_NAME_CONVERT, *1-189*
performance tuning, *1-2*
PERMIT_92_WRAP_FORMAT, *1-190*
PGA_AGGREGATE_LIMIT, *1-190*
PGA_AGGREGATE_TARGET, *1-191*
PLSCOPE_SETTINGS, *1-192*
PLSQL_CCFLAGS, *1-193*
PLSQL_CODE_TYPE, *1-194*
PLSQL_DEBUG, *1-194*
PLSQL_OPTIMIZE_LEVEL, *1-195*
PLSQL_V2_COMPATIBILITY, *1-196*
PLSQL_WARNINGS, *1-197*
PRE_PAGE_SGA, *1-198*
PROCESSES, *1-199*
PROCESSOR_GROUP_NAME, *1-199*
QUERY_REWRITE_ENABLED, *1-200*
QUERY_REWRITE_INTEGRITY, *1-201*
RDBMS_SERVER_DN, *1-202*
READ_ONLY_OPEN_DELAYED, *1-203*
RECOVERY_PARALLELISM, *1-203*
RECYCLEBIN, *1-204*
REDO_TRANSPORT_USER, *1-204*
initialization parameters *(continued)*
REMOTE_DEPENDENCIES_MODE, *1-205*
REMOTE_LISTENER, *1-206*
REMOTE_LOGIN_PASSWORDFILE, *1-206*
REMOTE_OS_AUTHENT, *1-208*
REMOTE_OS_ROLES, *1-208*
REPLICATION_DEPENDENCY_TRACKING, *1-209*
RESOURCE_LIMIT, *1-209*
RESOURCE_MANAGER_CPU_ALLOCATION, *1-210*
RESOURCE_MANAGER_PLAN, *1-210*
RESULT_CACHE_MAX_RESULT, *1-212*
RESULT_CACHE_MAX_SIZE, *1-213*
RESULT_CACHE_MODE, *1-213*
RESULT_CACHE_REMOTE_EXPIRATION, *1-214*
RESUMABLE_TIMEOUT, *1-215*
ROLLBACK_SEGMENTS, *1-215*
SEC_CASE_SENSITIVE_LOGON, *1-216*
SEC_MAX_FAILED_LOGIN_ATTEMPTS, *1-217*
SEC_PROTOCOL_ERROR_FURTHER_ACTION, *1-217*
SEC_PROTOCOL_ERROR_TRACE_ACTION, *1-218*
SEC_RETURN_SERVER_RELEASE_BANNER, *1-218*
SERIAL_REUSE, *1-219*
SERVICE_NAMES, *1-220*
SESSION_CACHED_CURSORS, *1-221*
SESSION_MAX_OPEN_FILES, *1-222*
SESSIONS, *1-222*
SGA_MAX_SIZE, *1-223*
SGA_TARGET, *1-224*
SHADOW_CORE_DUMP, *1-225*
SHARED_MEMORY_ADDRESS, *1-226*
SHARED_POOL_RESERVED_SIZE, *1-226*
SHARED_POOL_SIZE, *1-227*
SHARED_SERVER_SESSIONS, *1-228*
SHARED_SERVERS, *1-228*
SKIP_UNUSABLE_INDEXES, *1-229*
SMTP_OUT_SERVER, *1-230*
SORT_AREA_RETAINED_SIZE, *1-231*
SORT_AREA_SIZE, *1-232*
SPATIAL_VECTOR_ACCELERATION, *1-233*
specifying in initialization parameter files, *1-4*
SPFILE, *1-233*
SQL_TRACE, *1-234*
SQL92_SECURITY, *1-235*
SQLTUNE_CATEGORY, *1-236*
STANDBY_ARCHIVE_DEST, *1-236*
STANDBY_FILE_MANAGEMENT, *1-237*
STAR_TRANSFORMATION_ENABLED, *1-237*
STATISTICS_LEVEL, *1-238*
STREAMS_POOL_SIZE, *1-240*
TAPE_ASYNCH_IO, *1-241*
TEMP_UNDO_ENABLED, *1-241*

initialization parameters *(continued)*
THREAD, *1-242*
THREADED_EXECUTION, *1-242*
TIMED_OS_STATISTICS, *1-244*
TIMED_STATISTICS, *1-244*
TRACE_ENABLED, *1-245*
TRACEFILE_IDENTIFIER, *1-246*
TRANSACTIONS, *1-247*
TRANSACTIONS_PER_ROLLBACK_SEGMENT, *1-247*
UNDO_MANAGEMENT, *1-248*
UNDO_RETENTION, *1-248*
UNDO_TABLESPACE, *1-249*
UNIFIED_AUDIT_SGA_QUEUE_SIZE, *1-250*
USE_DEDICATED_BROKER, *1-250*
USE_INDIRECT_DATA_BUFFERS, *1-252*
USE_LARGE_PAGES, *1-252*
USER_DUMP_DEST, *1-253*
UTL_FILE_DIR, *1-254*
variable, *1-2*
WORKAREA_SIZE_POLICY, *1-255*
XML_DB_EVENTS, *1-255*
INITJVM.SQL script, *B-6*
INMEMORY_CLAUSE_DEFAULT initialization parameter, *1-100*
INMEMORY_FORCE initialization parameter, *1-105*
INMEMORY_MAX_POPULATE_SERVERS initialization parameter, *1-106*
INMEMORY_QUERY initialization parameter, *1-107*
INMEMORY_SIZE initialization parameter, *1-108*
INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT initialization parameter, *1-109*
inserts
locating free space, *1-111*
INST_ID column, *7-2*
INSTANCE clause
of ALTER TABLE
allocating extents, *1-111*
INSTANCE_GROUPS initialization parameter, *1-110*
INSTANCE_NAME initialization parameter, *1-110*
INSTANCE_NUMBER initialization parameter, *1-111*
INSTANCE_TYPE initialization parameter, *1-112*
instances
checkpoint, *1-138*
startup order, *1-111*

## J

Java initialization parameters
JAVA_MAX_SESSIONSPACE_SIZE, *1-113*
JAVA_POOL_SIZE, *1-113*
JAVA_SOFT_SESSIONSPACE_LIMIT, *1-114*
Java scripts, *B-6*
JAVA_JIT_ENABLED initialization parameter, *1-112*
JAVA_MAX_SESSIONSPACE_SIZE initialization parameter, *1-113*
JAVA_POOL_SIZE initialization parameter, *1-113*
JAVA_SOFT_SESSIONSPACE_LIMIT initialization parameter, *1-114*
JOB_QUEUE_PROCESSES initialization parameter, *1-115*

## L

language
database default language, *1-154*
LARGE_POOL_SIZE initialization parameter, *1-115*
LDAP_DIRECTORY_ACCESS initialization parameter, *1-117*
LDAP_DIRECTORY_SYSAUTH initialization parameter, *1-118*
LICENSE_MAX_SESSIONS initialization parameter, *1-118*
LICENSE_MAX_USERS initialization parameter, *1-119*
LICENSE_SESSIONS_WARNING initialization parameter, *1-120*
licenses
maximum sessions, *1-119*
maximum users, *1-119*
warning for concurrent user sessions, *1-120*
limits on the database, *A-1*
linguistic sorts, *1-158*
LISTENER_NETWORKS initialization parameter, *1-121*
local currency, *1-149*, *1-151*
LOCAL_LISTENER initialization parameter, *1-121*
LOCK_NAME_SPACE initialization parameter, *1-122*
LOCK_SGA initialization parameter, *1-122*
locks
names, *C-4*
LOG_ARCHIVE_CONFIG initialization parameter, *1-123*
LOG_ARCHIVE_DEST initialization parameter, *1-124*
LOG_ARCHIVE_DEST_n initialization parameter, *1-125*
LOG_ARCHIVE_DEST_STATE_n initialization parameter, *1-129*
LOG_ARCHIVE_DUPLEX_DEST initialization parameter, *1-130*
LOG_ARCHIVE_FORMAT initialization parameter, *1-131*
LOG_ARCHIVE_MAX_PROCESSES initialization parameter, *1-133*
LOG_ARCHIVE_MIN_SUCCEED_DEST initialization parameter, *1-133*
LOG_ARCHIVE_TRACE initialization parameter, *1-134*
LOG_BUFFER initialization parameter, *1-137*
LOG_CHECKPOINT_INTERVAL initialization parameter, *1-138*
LOG_CHECKPOINT_TIMEOUT initialization parameter, *1-139*
LOG_CHECKPOINTS_TO_ALERT initialization parameter, *1-139*

LOG_FILE_NAME_CONVERT initialization parameter, *1-140*
LOGSTDBY_UNSUPPORTED_TABLES synonym for DBA_LOGSTDBY_UNSUPPORTED_TABLE view, *6-104*
lowercase significance, *1-4*

## M

MAP_OBJECT table, *6-104*
MAX_DISPATCHERS initialization parameter, *1-141*
MAX_DUMP_FILE_SIZE initialization parameter, *1-141*
MAX_ENABLED_ROLES initialization parameter, *1-142*
MAX_SHARED_SERVERS initialization parameter, *1-143*
MAX_STRING_SIZE initialization parameter, *1-144*
memory
  amount used for sorting, *1-231*
  virtual, *1-2*
MEMORY_MAX_TARGET initialization parameter, *1-147*
MEMORY_TARGET initialization parameter, *1-147*
monitor
  performance, *1-138*
multiple-process systems
  number of processes, *1-199*
multitenant container databases *See* CDBs.
  *See* CDBs.

## N

NLS_CALENDAR initialization parameter, *1-148*
NLS_COMP initialization parameter, *1-149*
NLS_CURRENCY initialization parameter
  defined by NLS_TERRITORY, *1-159*
NLS_DATABASE_PARAMETERS view, *6-105*
NLS_DATE_FORMAT initialization parameter
  defined by NLS_NUMERIC_CHARACTERS, *1-157*
  defined by NLS_TERRITORY, *1-159*
NLS_DATE_LANGUAGE initialization parameter, *1-152*
NLS_DUAL_CURRENCY initialization parameter, *1-153*
NLS_INSTANCE_PARAMETERS view, *6-105*
NLS_ISO_CURRENCY initialization parameter
  defined by NLS_TERRITORY, *1-159*
NLS_LANGUAGE initialization parameter, *1-154*
NLS_LENGTH_SEMANTICS initialization parameter, *1-155*
NLS_NCHAR_CONV_EXCP initialization parameter, *1-156*
NLS_NUMERIC_CHARACTERS initialization parameter, *1-157*
NLS_SESSION_PARAMETERS view, *6-106*
NLS_SORT initialization parameter, *1-157*
NLS_TERRITORY initialization parameter, *1-159*
NLS_TIME_FORMAT, *8-81*
NLS_TIME_TZ_FORMAT, *8-81*
NLS_TIMESTAMP_FORMAT initialization parameter, *1-159*
NLS_TIMESTAMP_TZ_FORMAT initialization parameter, *1-160*
NONCDB_COMPATIBLE initialization parameter, *1-161*
numeric group separators, *1-157*

## O

O7_DICTIONARY_ACCESSIBILITY initialization parameter, *1-162*
OBJ synonym for USER_OBJECTS view, *6-106*
OBJECT_CACHE_MAX_SIZE_PERCENT initialization parameter, *1-163*
OBJECT_CACHE_OPTIMAL_SIZE initialization parameter, *1-163*
objects
  data dictionary, *2-2*
OLAP_PAGE_POOL_SIZE initialization parameter, *1-164*
online redo log
  archiving mode, *1-124*
  block, *1-138*
  file size, *1-138*
  setting checkpoint interval, *1-138*
OPEN_CURSORS initialization parameter, *1-164*
OPEN_LINKS initialization parameter, *1-165*
OPEN_LINKS_PER_INSTANCE initialization parameter, *1-166*
operating system
  authenticating remote clients, *1-208*
  authenticating users and passwords, *1-176*
  dependent parameters, *1-2*
  roles for remote clients, *1-208*
optimization
  trading cursor space for time, *1-53*
OPTIMIZER_ADAPTIVE_FEATURES initialization parameter, *1-166*
OPTIMIZER_ADAPTIVE_REPORTING_ONLY initialization parameter, *1-167*
OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES initialization parameter, *1-167*
OPTIMIZER_DYNAMIC_SAMPLING initialization parameter, *1-168*
OPTIMIZER_FEATURES_ENABLE initialization parameter, *1-169*
OPTIMIZER_INDEX_CACHING initialization parameter, *1-171*
OPTIMIZER_INDEX_COST_ADJ initialization parameter, *1-172*
OPTIMIZER_INMEMORY_AWARE initialization parameter, *1-172*
OPTIMIZER_MODE initialization parameter, *1-173*

OPTIMIZER_SECURE_VIEW_MERGING initialization parameter, *1-174*
OPTIMIZER_USE_INVISIBLE_INDEXES initialization parameter, *1-175*
OPTIMIZER_USE_PENDING_STATISTICS initialization parameter, *1-175*
OPTIMIZER_USE_SQL_PLAN_BASELINES initialization parameter, *1-175*
Oracle Database Real Application Security views, *2-2*
Oracle Database Vault views, *2-3*
Oracle Label Security views, *2-3*
Oracle Replication views, *2-3*
Oracle Workspace Manager views, *2-5*
ORPHAN_KEY_TABLE table *See* DBA_ORPHAN_KEY_TABLE view
  *See* DBA_ORPHAN_KEY_TABLE view
OS_AUTHENT_PREFIX initialization parameter, *1-176*
OS_ROLES initialization parameter, *1-177*

## P

parallel query
  maximum number of instances, *1-181*
  maximum number of servers, *1-184*
  minimum number of queries, *1-186*
PARALLEL_ADAPTIVE_MULTI_USER initialization parameter, *1-177*
PARALLEL_AUTOMATIC_TUNING initialization parameter, *1-178*
PARALLEL_DEGREE_LIMIT initialization parameter, *1-179*
PARALLEL_DEGREE_POLICY initialization parameter, *1-180*
PARALLEL_EXECUTION_MESSAGE_SIZE initialization parameter, *1-181*
PARALLEL_FORCE_LOCAL initialization parameter, *1-182*
PARALLEL_INSTANCE_GROUP initialization parameter, *1-182*
PARALLEL_IO_CAP_ENABLED initialization parameter, *1-183*
PARALLEL_MAX_SERVERS initialization parameter, *1-184*
PARALLEL_MIN_PERCENT initialization parameter, *1-185*
PARALLEL_MIN_SERVERS initialization parameter, *1-186*
PARALLEL_MIN_TIME_THRESHOLD initialization parameter, *1-187*
PARALLEL_SERVERS_TARGET initialization parameter, *1-187*
PARALLEL_THREADS_PER_CPU initialization parameter, *1-188*
passwords
  authenticating, *1-176*, *1-208*
  REMOTE_LOGIN_PASSWORDFILE initialization parameter, *1-206*
PATH_VIEW view, *6-106*
PDB_ALERTS view, *6-106*
PDB_FILE_NAME_CONVERT initialization parameter, *1-189*
PDB_PLUG_IN_VIOLATIONS view, *6-107*
PDBs, *1-28*
performance
  shared pool, *1-227*
PERMIT_92_WRAP_FORMAT initialization parameter, *1-190*
PGA_AGGREGATE_LIMIT initialization parameter, *1-190*
PGA_AGGREGATE_TARGET initialization parameter, *1-191*
PLAN_TABLE table, *6-108*
PLSCOPE_SETTINGS initialization parameter, *1-192*
PLSQL_CCFLAGS initialization parameter, *1-193*
PLSQL_CODE_TYPE initialization parameter, *1-194*
PLSQL_DEBUG initialization parameter, *1-194*
PLSQL_OPTIMIZE_LEVEL initialization parameter, *1-195*
PLSQL_V2_COMPATIBILITY initialization parameter, *1-196*
PLSQL_WARNINGS initialization parameter, *1-197*
pluggable databases *See* PDBs.
  *See* PDBs.
PLUGGABLE_SET_CHECK view, *6-111*
PRE_PAGE_SGA initialization parameter, *1-198*
privileges
  remote login, *1-206*
  RESTRICTED_SESSION privilege, *1-119*, *1-120*
  table-level select privileges, *1-235*
procedures
  shared pool, *1-227*
processes
  dispatcher process maximum number, *1-141*
  maximum shared server processes, *1-143*
  number of server processes, *1-113*, *1-114*, *1-228*, *1-229*
  recovery, *1-204*
  trace files, *1-253*
  user processes, *1-199*
PROCESSES initialization parameter, *1-199*
PROCESSOR_GROUP_NAME initialization parameter, *1-199*
PRODUCT_COMPONENT_VERSION view, *6-112*
profiles
  resource limits, *1-209*, *1-212*
PROXY_USERS view, *6-112*
PSTUBTBL table, *6-113*
PUBLIC clause
  of ALTER DATABASE
    thread of redo, *1-242*
PUBLIC_DEPENDENCY view, *6-113*
PUBLICSYN view, *6-113*

## Q


queries
  hints, *1-173*
QUERY_REWRITE_ENABLED initialization parameter, *1-200*
QUERY_REWRITE_INTEGRITY initialization parameter, *1-201*
QUEUE_PRIVILEGES view, *6-114*


## R


RC_ARCHIVED_LOG view, *2-6*
RC_BACKUP_ARCHIVELOG_DETAILS view, *2-6*
RC_BACKUP_ARCHIVELOG_SUMMARY view, *2-6*
RC_BACKUP_CONTROLFILE view, *2-6*
RC_BACKUP_CONTROLFILE_DETAILS view, *2-6*
RC_BACKUP_CONTROLFILE_SUMMARY view, *2-6*
RC_BACKUP_COPY_DETAILS view, *2-6*
RC_BACKUP_COPY_SUMMARY view, *2-6*
RC_BACKUP_CORRUPTION view, *2-6*
RC_BACKUP_DATAFILE view, *2-6*
RC_BACKUP_DATAFILE_DETAILS view, *2-6*
RC_BACKUP_DATAFILE_SUMMARY view, *2-6*
RC_BACKUP_FILES view, *2-6*
RC_BACKUP_PIECE view, *2-6*
RC_BACKUP_PIECE_DETAILS view, *2-6*
RC_BACKUP_REDOLOG view, *2-6*
RC_BACKUP_SET view, *2-6*
RC_BACKUP_SET_DETAILS view, *2-6*
RC_BACKUP_SET_SUMMARY view, *2-6*
RC_BACKUP_SPFILE view, *2-6*
RC_BACKUP_SPFILE_DETAILS view, *2-6*
RC_BACKUP_SPFILE_SUMMARY view, *2-6*
RC_CHECKPOINT view, *2-6*
RC_CONTROLFILE_COPY view, *2-6*
RC_COPY_CORRUPTION view, *2-6*
RC_DATABASE view, *2-6*
RC_DATABASE_BLOCK_CORRUPTION view, *2-7*
RC_DATABASE_INCARNATION view, *2-7*
RC_DATAFILE view, *2-7*
RC_DATAFILE_COPY view, *2-7*
RC_LOG_HISTORY view, *2-7*
RC_OFFLINE_RANGE view, *2-7*
RC_PROXY_ARCHIVEDLOG view, *2-7*
RC_PROXY_ARCHIVELOG_DETAILS view, *2-7*
RC_PROXY_ARCHIVELOG_SUMMARY view, *2-7*
RC_PROXY_CONTROLFILE view, *2-7*
RC_PROXY_COPY_DETAILS view, *2-7*
RC_PROXY_COPY_SUMMARY view, *2-7*
RC_PROXY_DATAFILE view, *2-7*
RC_REDO_LOG view, *2-7*
RC_REDO_THREAD view, *2-7*
RC_RESTORE_POINT view, *2-7*
RC_RESYNC view, *2-7*
RC_RMAN_BACKUP_JOB_DETAILS view, *2-7*
RC_RMAN_BACKUP_SUBJOB_DETAILS view, *2-7*
RC_RMAN_BACKUP_TYPE view, *2-7*
RC_RMAN_CONFIGURATION view, *2-7*
RC_RMAN_OUTPUT view, *2-7*
RC_RMAN_STATUS view, *2-7*
RC_SITE view, *2-7*
RC_STORED_SCRIPT view, *2-7*
RC_STORED_SCRIPT_LINE view, *2-7*
RC_TABLESPACE view, *2-7*
RC_TEMPFILE view, *2-7*
RC_UNUSABLE_BACKUPFILE_DETAILS view, *2-7*
RDBMS_SERVER_DN initialization parameter, *1-202*
READ_ONLY_OPEN_DELAYED initialization parameter, *1-203*
Real Application Clusters
  dynamic performance views, *7-2*
    initialization parameters
      CIRCUITS, *1-37*
      CLUSTER_DATABASE, *1-39*
      CLUSTER_DATABASE_INSTANCES, *1-39*
      CLUSTER_INTERCONNECTS, *1-40*
      DISPATCHERS, *1-82*
      FAST_START_PARALLEL_ROLLBACK, *1-92*
      PARALLEL_ADAPTIVE_MULTI_USER, *1-178*
      PARALLEL_AUTOMATIC_TUNING, *1-178*
      PARALLEL_EXECUTION_MESSAGE_SIZE, *1-181*
      PARALLEL_INSTANCE_GROUP, *1-182*
      PARALLEL_MAX_SERVERS, *1-184*
      PARALLEL_MIN_PERCENT, *1-185*
      PARALLEL_MIN_SERVERS, *1-186*
      PARALLEL_THREADS_PER_CPU, *1-188*
recovery
  number of required processes, *1-204*
Recovery Catalog views, *2-6*
RECOVERY_PARALLELISM initialization parameter, *1-203*
RECYCLEBIN initialization parameter, *1-204*
RECYCLEBIN synonym for USER_RECYCLEBIN view, *6-114*
REDACTION_COLUMNS view, *6-114*
REDACTION_POLICIES view, *6-115*
REDACTION_VALUES_FOR_TYPE_FULL view, *6-116*
redo logs
  buffer size, *1-137*
redo thread, *1-242*
REDO_TRANSPORT_USER initialization parameter, *1-204*
remote clients
  authenticating, *1-208*
remote logins, *1-206*
REMOTE_DEPENDENCIES_MODE initialization parameter, *1-205*
REMOTE_LISTENER initialization parameter, *1-206*

REMOTE_LOGIN_PASSWORDFILE initialization parameter, *1-206*
REMOTE_OS_AUTHENT initialization parameter, *1-208*
REMOTE_OS_ROLES initialization parameter, *1-208*
REPAIR_TABLE table *See* DBA_REPAIR_TABLE view
*See* DBA_REPAIR_TABLE view
REPLICATION_DEPENDENCY_TRACKING initialization parameter, *1-209*
REPORT_COMPONENTS view, *6-117*
REPORT_FILES view, *6-117*
REPORT_FORMATS view, *6-117*
resource limits for profiles, *1-209*, *1-212*
RESOURCE_COST view, *6-118*
RESOURCE_LIMIT initialization parameter, *1-209*
RESOURCE_MANAGER_CPU_ALLOCATION initialization parameter, *1-210*
RESOURCE_MANAGER_PLAN initialization parameter, *1-210*
RESOURCE_MAP table, *6-118*
RESOURCE_VIEW view, *6-119*
RESULT_CACHE_MAX_RESULT initialization parameter, *1-212*
RESULT_CACHE_MAX_SIZE initialization parameter, *1-213*
RESULT_CACHE_MODE initialization parameter, *1-213*
RESULT_CACHE_REMOTE_EXPIRATION initialization parameter, *1-214*
RESUMABLE_TIMEOUT initialization parameter, *1-215*
RMJVM.SQL script, *B-6*
ROLE_ROLE_PRIVS view, *6-119*
ROLE_SYS_PRIVS view, *6-119*
ROLE_TAB_PRIVS view, *6-120*
ROLE_WM_PRIVS view, *2-6*
roles
remote clients, *1-208*
ROLLBACK_SEGMENTS initialization parameter, *1-215*

## S

SCHEDULER_BATCH_ERRORS view, *6-120*
SCHEMA_EXPORT_OBJECTS view, *6-121*
scripts
CATALOG.SQL, *B-2*
CATBLOCK.SQL, *B-2*
CATCLUST.SQL, *B-2*
CATHS.SQL, *B-2*
CATJAVA.SQL, *B-6*
CATNOCLUST.SQL, *B-5*
CATNOJAV.SQL, *B-5*
CATNOPRT.SQL, *B-5*
CATNOSVM.SQL, *B-5*
CATNSNMP.SQL, *B-6*
CATPROC.SQL, *B-2*
scripts *(continued)*
CATWRR.SQL, *B-3*
data dictionary, *B-2*
DBMSPOOL.SQL, *B-3*
INITJVM.SQL, *B-6*
Java, *B-6*
RMJVM.SQL, *B-6*
SQL scripts, *B-1*
upgrading, *B-6*
UTLCHN1.SQL, *B-3*
UTLEXPT1.SQL, *B-3*
UTLIRP.SQL, *B-4*
UTLLOCKT.SQL, *B-4*
UTLPWDMG.SQL, *B-4*
UTLRP.SQL, *B-4*
UTLXPLAN.SQL, *B-5*
SEC_CASE_SENSITIVE_LOGON initialization parameter, *1-216*
SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter, *1-217*
SEC_PROTOCOL_ERROR_FURTHER_ACTION initialization parameter, *1-217*
SEC_PROTOCOL_ERROR_TRACE_ACTION initialization parameter, *1-218*
SEC_RETURN_SERVER_RELEASE_BANNER initialization parameter, *1-218*
SEQ synonym for USER_SEQUENCES view, *6-121*
SERIAL_REUSE initialization parameter, *1-219*
server parameter file
overview, *1-3*
server processes
number, *1-113*, *1-114*, *1-228*, *1-229*
SERVICE_NAMES initialization parameter, *1-220*
SESSION_CACHED_CURSORS initialization parameter, *1-221*
SESSION_CONTEXT view, *6-122*
SESSION_MAX_OPEN_FILES initialization parameter, *1-222*
SESSION_PRIVS view, *6-122*
SESSION_ROLES view, *6-122*
SESSIONS initialization parameter, *1-222*
SGA_MAX_SIZE initialization parameter, *1-223*
SGA_TARGET initialization parameter, *1-224*
SHADOW_CORE_DUMP initialization parameter, *1-225*
shared server
SESSIONS initialization parameter, *1-222*
shared server processes
maximum number, *1-143*
SHARED_MEMORY_ADDRESS initialization parameter, *1-226*
SHARED_POOL_RESERVED_SIZE initialization parameter, *1-226*
SHARED_POOL_SIZE initialization parameter, *1-227*
SHARED_SERVER_SESSIONS initialization parameter, *1-228*
SHARED_SERVERS initialization parameter, *1-228*

SHOW PARAMETERS command, 1-21
SKIP_UNUSABLE_INDEXES initialization parameter, 1-229
SMTP_OUT_SERVER initialization parameter, 1-230
SORT_AREA_RETAINED_SIZE initialization parameter, 1-231
SORT_AREA_SIZE initialization parameter, 1-232
sorting
  maximum memory, 1-231
  ORDER BY queries, 1-158
SOURCE_SIZE view, 6-122
SPATIAL_VECTOR_ACCELERATION initialization parameter, 1-233
SPFILE initialization parameter, 1-233
SQL scripts
  creating the data dictionary, B-1
  optional, B-2
  required, B-1
SQL_TRACE initialization parameter, 1-234
SQL92_SECURITY initialization parameter, 1-235
SQLTUNE_CATEGORY initialization parameter, 1-236
STANDBY_ARCHIVE_DEST initialization parameter, 1-236
STANDBY_FILE_MANAGEMENT initialization parameter, 1-237
STAR_TRANSFORMATION_ENABLED initialization parameter, 1-237
starting up
  instance number, 1-111
  startup order, 1-111
statistics
  application wait time, E-2
  background checkpoints completed, E-2
  background checkpoints started, E-2
  background timeouts, E-2
  branch node splits, E-2
  buffer is not pinned count, E-2
  buffer is pinned count, E-2
  bytes received via SQL*Net from client, E-2
  bytes received via SQL*Net from dblink, E-2
  bytes sent via SQL*Net to client, E-2
  bytes sent via SQL*Net to dblink, E-2
  Cached Commit SCN referenced, E-3
    calls to get snapshot scn kcmgss, E-3
  calls to kcmgas, E-3
  calls to kcmgcs, E-3
  calls to kcmgrs, E-3
  change write time, E-3
  cleanouts and rollbacks - consistent read gets, E-3
  cleanouts only - consistent read gets, E-3
  cluster key scan block gets, E-3
  cluster key scans, E-3
  cluster wait time, E-3
  cold recycle reads, E-3
    commit cleanout failures
statistics (continued)
    commit cleanout failures (continued)
      block lost, E-3
      buffer being written, E-3
      callback failure, E-3
      cannot pin, E-3
      hot backup in progress, E-3
      write disabled, E-3
  commit cleanouts, E-4
  commit cleanouts successfully completed, E-4
  commit nowait performed, E-4
  commit nowait requested, E-4
  Commit SCN cached, E-4
  commit wait performed, E-4
  commit wait requested, E-4
  commit wait/nowait requested, E-4
  concurrency wait time, E-4
  consistent changes, E-4
  consistent gets, E-4
  consistent gets direct, E-4
  consistent gets from cache, E-4
  CPU used by this session, E-5
  CPU used when call started, E-5
  CR blocks created, E-5
  current blocks converted for CR, E-5
  cursor authentications, E-5
  data blocks consistent reads - undo records applied, E-5
  data warehousing cooling action, E-5
  data warehousing evicted objects, E-5
  data warehousing evicted objects - cooling, E-5
  data warehousing evicted objects - replace, E-5
  data warehousing scanned blocks, E-5
  data warehousing scanned blocks - disk, E-5
  data warehousing scanned blocks - memory, E-5
  data warehousing scanned blocks - offload, E-5
  data warehousing scanned objects, E-5
  db block changes, E-5
  db block gets, E-6
  db block gets direct, E-6
  db block gets from cache, E-6
  DBWR checkpoint buffers written, E-6
  DBWR checkpoints, 1-138, E-6
  DBWR lru scans, E-6
  DBWR revisited being-written buffer, E-6
  DBWR transaction table writes, E-6
  DBWR undo block writes, E-6
  DDL statements parallelized, E-6
  deferred (CURRENT) block cleanout applications, E-6
  DFO trees parallelized, E-6
  dirty buffers inspected, E-7
  DML statements parallelized, E-7
  enqueue conversions, E-7
  enqueue deadlocks, E-7
  enqueue releases, E-7

statistics *(continued)*
enqueue requests, *E-7*
enqueue timeouts, *E-7*
enqueue waits, *E-7*
exchange deadlocks, *E-7*
execute count, *E-7*
fdba woken up, *E-7*
file io wait time, *E-7*
flash cache eviction: aged out, *E-7*
flash cache eviction: buffer pinned, *E-7*
flash cache eviction: nvalidated, *E-7*
flash cache insert skip: corrupt, *E-7*
flash cache insert skip: DBWR overloaded, *E-8*
flash cache insert skip: exists, *E-8*
flash cache insert skip: modification, *E-8*
flash cache insert skip: not current, *E-8*
flash cache insert skip: not useful, *E-8*
flash cache inserts, *E-8*
flashback log write bytes, *E-8*
flashback log writes, *E-8*
foreground propagated tracked transactions, *E-8*
free buffer inspected, *E-8*
free buffer requested, *E-8*
gc read wait failure, *E-8*
gc read wait time, *E-8*
gc read wait timeouts, *E-8*
gc read waits, *E-9*
global cache blocks corrupt, *E-9*
global cache convert time, *E-9*
global cache convert timeouts, *E-9*
global cache converts, *E-9*
global cache cr block log flush time, *E-9*
global cache cr block receive time, *E-9*
global cache cr block send time, *E-9*
global cache cr block serve time, *E-9*
global cache cr blocks served, *E-9*
global cache cr requests blocked, *E-9*
global cache cr timeouts, *E-9*
global cache defers, *E-9*
global cache freelist waits, *E-9*
global cache get time, *E-10*
global cache gets, *E-10*
global cache prepare failures, *E-10*
global enqueue CPU used by this session, *E-10*
global enqueue get time, *E-10*
global enqueue gets async, *E-10*
global enqueue gets sync, *E-10*
global enqueue releases, *E-10*
hot buffers moved to head of LRU, *E-10*
immediate (CR) block cleanout applications, *E-10*
immediate (CURRENT) block cleanout applications, *E-10*
in call idle wait time, *E-10*
index fast full scans (direct read), *E-10*
index fast full scans (full), *E-10*
index fast full scans (rowid ranges), *E-10*
statistics *(continued)*
instance recovery database freeze count, *E-10*
kcmccs called get current scn, *E-10*
kcmccs read scn without going to DLM, *E-11*
kcmggs waited for batching, *E-11*
large tracked transactions, *E-11*
leaf node splits, *E-11*
lob reads, *E-11*
lob writes, *E-11*
lob writes unaligned, *E-11*
logons cumulative, *E-11*
logons current, *E-11*
messages received, *E-11*
messages sent, *E-11*
native hash arithmetic execute, *E-11*
native hash arithmetic fail, *E-11*
next scns gotten without going to DLM, *E-11*
no buffer to keep pinned count, *E-11*
no work - consistent read gets, *E-12*
non-idle wait count, *E-12*
non-idle wait time, *E-12*
OLAP Aggregate Function Calc, *E-12*
OLAP Aggregate Function Logical NA, *E-12*
OLAP Aggregate Function Precompute, *E-12*
OLAP Custom Member Limit, *E-13*
OLAP Engine Calls, *E-13*
OLAP Fast Limit, *E-13*
OLAP Full Limit, *E-13*
OLAP GID Limit, *E-13*
OLAP Import Rows Loaded, *E-13*
OLAP Import Rows Pushed, *E-12*
OLAP INHIER Limit, *E-13*
OLAP Limit Time, *E-13*
OLAP Paging Manager Cache Changed Page, *E-13*
OLAP Paging Manager Cache Hit, *E-12*
OLAP Paging Manager Cache Miss, *E-12*
OLAP Paging Manager Cache Write, *E-12*
OLAP Paging Manager New Page, *E-12*
OLAP Paging Manager Pool Size, *E-12*
OLAP Perm LOB Read, *E-13*
OLAP Row Id Limit, *E-13*
OLAP Row Load Time, *E-13*
OLAP Row Source Rows Processed, *E-13*
OLAP Session Cache Hit, *E-12*
OLAP Session Cache Miss, *E-12*
OLAP Temp Segments, *E-13*
OLAP Temp Segments Read, *E-13*
OLAP Unique Key Attribute Limit, *E-14*
opened cursors cumulative, *E-14*
opened cursors current, *E-14*
opens of replaced files, *E-14*
opens requiring cache replacement, *E-14*
operating system
OS All other sleep time, *E-14*
OS Chars read and written, *E-14*

statistics *(continued)*
operating system *(continued)*
OS CPU Qt wait time, *E-14*
OS Data page fault sleep time, *E-14*
OS Input blocks, *E-14*
OS Involuntary context switches, *E-14*
OS Kernel page fault sleep time, *E-14*
OS Major page faults, *E-15*
OS Messages received, *E-15*
OS Messages sent, *E-15*
OS Minor page faults, *E-15*
OS Other system trap CPU time, *E-15*
OS Output blocks, *E-15*
OS Process heap time, *E-15*
OS Signals received, *E-15*
OS Swaps, *E-15*
OS System call CPU time, *E-15*
OS System calls, *E-15*
OS Text page fault sleep time, *E-15*
OS User level CPU time, *E-15*
OS Voluntary context switches, *E-15*
Process stack size, *E-15*
Parallel operations downgraded 1 to 25 pct, *E-15*
Parallel operations downgraded 25 to 50 pct, *E-15*
Parallel operations downgraded 50 to 75 pct, *E-15*
Parallel operations downgraded 75 to 99 pct, *E-15*
Parallel operations downgraded to serial, *E-15*
Parallel operations not downgraded, *E-15*
parse count (describe), *E-16*
parse count (hard), *E-16*
parse count (total), *E-16*
parse time cpu, *E-16*
parse time elapsed, *E-16*
physical read bytes, *E-16*
physical read flash cache hits, *E-16*
physical read IO requests, *E-16*
physical read requests optimized, *E-16*
physical read total bytes, *E-16*
physical read total IO requests, *E-16*
physical read total multi block requests, *E-16*
physical reads, *E-16*
physical reads cache, *E-17*
physical reads cache prefetch, *E-17*
physical reads direct, *E-17*
physical reads direct (lob), *E-17*
physical reads direct temporary tablespace, *E-17*
physical reads for flashback new, *E-17*
physical reads prefetch warmup, *E-17*
physical write bytes, *E-17*
physical write IO requests, *E-17*
physical write total bytes, *E-17*
physical write total IO requests, *E-17*
physical write total multi block requests, *E-17*
physical writes, *E-17*
physical writes direct, *E-17*
physical writes direct (lob), *E-18*
statistics *(continued)*
physical writes direct temporary tablespace, *E-18*
physical writes from cache, *E-18*
physical writes non checkpoint, *E-18*
pinned buffers inspected, *E-18*
prefetched blocks aged out before use, *E-18*
process last non-idle time, *E-18*
PX local messages recv'd, *E-18*
PX local messages sent, *E-18*
PX remote messages recv'd, *E-18*
PX remote messages sent, *E-18*
queries parallelized, *E-18*
recovery array read time, *E-18*
recovery array reads, *E-18*
recovery blocks read, *E-18*
recovery blocks read for lost write detection, *E-18*
recovery blocks skipped lost write checks, *E-18*
recursive calls, *E-19*
recursive cpu usage, *E-19*
redo blocks checksummed by FG (exclusive), *E-19*
redo blocks checksummed by LGWR, *E-19*
redo blocks written, *E-19*
redo buffer allocation retries, *E-19*
redo entries, *E-19*
redo entries for lost write detection, *E-19*
redo log space requests, *E-19*
redo log space wait time, *E-19*
redo ordering marks, *E-19*
redo size, *E-20*
redo size for lost write detection, *E-20*
redo synch time, *E-20*
redo synch writes, *E-20*
redo wastage, *E-20*
redo write time, *E-20*
redo writer latching time, *E-20*
redo writes, *E-20*
remote instance undo block writes, *E-20*
remote instance undo header writes, *E-20*
rollback changes - undo records applied, *E-20*
rollbacks only - consistent read gets, *E-20*
rows fetched via callback, *E-20*
scheduler wait time, *E-20*
SCN increments due to another database, *E-20*
serializable aborts, *E-21*
session connect time, *E-21*
session cursor cache count, *E-21*
session cursor cache hits, *E-21*
session logical reads, *E-21*
session pga memory, *E-21*
session pga memory max, *E-21*
session stored procedure space, *E-21*
session uga memory, *E-21*
session uga memory max, *E-21*
shared hash latch upgrades - no wait, *E-21*
shared hash latch upgrades - wait, *E-21*
shared io pool buffer get failure, *E-21*

statistics *(continued)*
shared io pool buffer get success, *E-21*
slave propagated tracked transactions, *E-21*
sorts (disk), *E-22*
sorts (memory), *E-22*
sorts (rows), *E-22*
SQL*Net roundtrips to/from client, *E-22*
SQL*Net roundtrips to/from dblink, *E-22*
summed dirty queue length, *E-22*
switch current to new buffer, *E-22*
table fetch by rowid, *E-22*
table fetch continued row, *E-22*
table scan blocks gotten, *E-22*
table scan rows gotten, *E-23*
table scans (cache partitions), *E-23*
table scans (direct read), *E-23*
table scans (long tables), *E-23*
table scans (rowid ranges), *E-23*
table scans (short tables), *E-23*
timed, *1-245*
total file opens, *E-23*
tracked rows, *E-23*
tracked transactions, *E-23*
transaction lock background get time, *E-23*
transaction lock background gets, *E-23*
transaction lock foreground requests, *E-23*
transaction lock foreground wait time, *E-23*
transaction rollbacks, *E-23*
transaction tables consistent read rollbacks, *E-23*
transaction tables consistent reads - undo records applied, *E-23*
Unnecessary process cleanup for SCN batching, *E-23*
user calls, *E-24*
user commits, *E-24*
user I/O wait time, *E-24*
user rollbacks, *E-24*
very large tracked transactions, *E-24*
write clones created in background, *E-24*
write clones created in foreground, *E-24*
STATISTICS_LEVEL initialization parameter, *1-238*
STMT_AUDIT_OPTION_MAP table, *6-122*
STREAMS_POOL_SIZE initialization parameter, *1-240*
switch redo log file, *1-138*
SYN synonym for USER_SYNONYMS view, *6-123*
SYNONYMS view, *6-123*
SYS_OBJECTS view, *6-123*
SYSCATALOG view, *6-124*
SYSFILES view, *6-124*
SYSSEGOBJ view, *6-124*
system global area
buffer areas, *1-137*
system performance, *1-2*
SYSTEM_PRIVILEGE_MAP table, *6-124*

## T

TAB view, *6-124*
TABLE_EXPORT_OBJECTS view, *6-125*
TABLE_PRIVILEGE_MAP table, *6-125*
tables
data dictionary, *2-2*
DML locks, *1-85*
locating free space, *1-111*
TABQUOTAS view, *6-125*
TABS synonym for USER_TABLES view, *6-125*
tape archiving destination, *1-124*
TAPE_ASYNCH_IO initialization parameter, *1-241*
TEMP_UNDO_ENABLED initialization parameter, *1-241*
territory, *1-153*, *1-159*
THREAD initialization parameter, *1-242*
THREADED_EXECUTION initialization parameter, *1-242*
TIMED_OS_STATISTICS initialization parameter, *1-244*
TIMED_STATISTICS initialization parameter
file read/write statistics, *7-196*, *9-139*
TO_CHAR function, *1-151*
TO_DATE function, *1-151*
trace files
destination, *1-253*
maximum size, *1-142*
TRACE_ENABLED initialization parameter, *1-245*
TRACEFILE_IDENTIFIER initialization parameter, *1-246*
transactions
data locking, *1-85*
TRANSACTIONS initialization parameter
DML locks, *1-85*
TRANSACTIONS_PER_ROLLBACK_SEGMENT initialization parameter, *1-247*
TRUSTED_SERVERS view, *6-126*
TS_PITR_CHECK view, *6-126*
TS_PITR_OBJECTS TO_BE_DROPPED view, *6-127*

## U

UNDO_MANAGEMENT initialization parameter, *1-248*
UNDO_RETENTION initialization parameter, *1-248*
UNDO_TABLESPACE initialization parameter, *1-249*
UNI_PLUGGABLE_SET_CHECK view, *6-128*
UNIFIED_AUDIT_SGA_QUEUE_SIZE initialization parameter, *1-250*
UNIFIED_AUDIT_TRAIL view, *6-128*
updates
locating free space, *1-111*
upgrading
scripts, *B-6*
uppercase characters, *1-4*
USABLE_EDITIONS view, *6-136*

USE_DEDICATED_BROKER initialization parameter, *1-250*
USE_INDIRECT_DATA_BUFFERS initialization parameter, *1-252*
USE_LARGE_PAGES initialization parameter, *1-252*
user processes
  trace files, *1-253*
USER_ADDM_FDG_BREAKDOWN view, *6-136*
USER_ADDM_FINDINGS view, *6-136*
USER_ADDM_INSTANCES view, *6-137*
USER_ADDM_TASK_DIRECTIVES view, *6-137*
USER_ADDM_TASKS view, *6-137*
USER_ADVISOR_ACTIONS view, *6-137*
USER_ADVISOR_DIR_TASK_INST view, *6-137*
USER_ADVISOR_EXEC_PARAMETERS view, *6-137*
USER_ADVISOR_EXECUTIONS view, *6-137*
USER_ADVISOR_FDG_BREAKDOWN view, *6-138*
USER_ADVISOR_FINDINGS view, *6-138*
USER_ADVISOR_JOURNAL view, *6-138*
USER_ADVISOR_LOG view, *6-138*
USER_ADVISOR_OBJECTS view, *6-138*
USER_ADVISOR_PARAMETERS view, *6-139*
USER_ADVISOR_RATIONALE view, *6-139*
USER_ADVISOR_RECOMMENDATIONS view, *6-139*
USER_ADVISOR_SQLA_REC_SUM view, *6-139*
USER_ADVISOR_SQLA_TABLES view, *6-140*
USER_ADVISOR_SQLA_WK_MAP view, *6-140*
USER_ADVISOR_SQLA_WK_STMTS view, *6-140*
USER_ADVISOR_SQLPLANS view, *6-140*
USER_ADVISOR_SQLSTATS view, *6-140*
USER_ADVISOR_SQLW_JOURNAL view, *6-140*
USER_ADVISOR_SQLW_PARAMETERS view, *6-141*
USER_ADVISOR_SQLW_STMTS view, *6-141*
USER_ADVISOR_SQLW_SUM view, *6-141*
USER_ADVISOR_SQLW_TABLES view, *6-141*
USER_ADVISOR_SQLW_TEMPLATES view, *6-142*
USER_ADVISOR_TASKS view, *6-142*
USER_ADVISOR_TEMPLATES view, *6-142*
USER_ALL_TABLES view, *6-142*
USER_APPLY_ERROR view, *6-142*
USER_AQ_AGENT_PRIVS view, *6-143*
USER_ARGUMENTS view, *6-143*
USER_ASSEMBLIES view, *6-143*
USER_ASSOCIATIONS view, *6-143*
USER_ATTRIBUTE_TRANSFORMATIONS view, *6-144*
USER_AUDIT_OBJECT view, *6-144*
USER_AUDIT_POLICIES view, *6-144*
USER_AUDIT_POLICY_COLUMNS view, *6-145*
USER_AUDIT_SESSION view, *6-145*
USER_AUDIT_STATEMENT view, *6-145*
USER_AUDIT_TRAIL view, *6-146*
USER_AW_PS view, *6-146*
USER_AWS view, *6-146*
USER_BASE_TABLE_MVIEWS view, *6-147*
USER_CATALOG view
USER_CATALOG view *(continued)*
  CAT synonym, *3-186*
USER_CHANGE_NOTIFICATION_REGS view, *6-147*
USER_CLU_COLUMNS view, *6-147*
USER_CLUSTER_HASH_EXPRESSIONS view, *6-147*
USER_CLUSTERING_DIMENSIONS view, *6-147*
USER_CLUSTERING_JOINS view, *6-148*
USER_CLUSTERING_KEYS view, *6-148*
USER_CLUSTERING_TABLES view, *6-148*
USER_CLUSTERS view
  CLU synonym, *3-187*
USER_CODE_ROLE_PRIVS view, *6-149*
USER_COL_COMMENTS view, *6-149*
USER_COL_PENDING_STATS view, *6-149*
USER_COL_PRIVS view, *6-150*
USER_COL_PRIVS_MADE view, *6-150*
USER_COL_PRIVS_RECD view, *6-150*
USER_COLL_TYPES view, *6-150*
USER_COMPARISON view, *6-150*
USER_COMPARISON_COLUMNS view, *6-150*
USER_COMPARISON_ROW_DIF view, *6-151*
USER_COMPARISON_SCAN view, *6-151*
USER_COMPARISON_SCAN_VALUES view, *6-151*
USER_CONS_COLUMNS view, *6-151*
USER_CONS_OBJ_COLUMNS view, *6-151*
USER_CONSTRAINTS view, *6-151*
USER_CQ_NOTIFICATION_QUERIES view, *6-151*
USER_CREDENTIALS view, *6-152*
USER_CUBE_ATTR_VISIBILITY view, *6-152*
USER_CUBE_ATTRIBUTES view, *6-152*
USER_CUBE_BUILD_PROCESSES view, *6-152*
USER_CUBE_CALCULATED_MEMBERS view, *6-152*
USER_CUBE_DIM_LEVELS view, *6-153*
USER_CUBE_DIM_MODELS view, *6-153*
USER_CUBE_DIM_VIEW_COLUMNS view, *6-153*
USER_CUBE_DIM_VIEWS view, *6-153*
USER_CUBE_DIMENSIONALITY view, *6-154*
USER_CUBE_DIMENSIONS view, *6-154*
USER_CUBE_HIER_LEVELS view, *6-154*
USER_CUBE_HIER_VIEW_COLUMNS view, *6-154*
USER_CUBE_HIER_VIEWS view, *6-154*
USER_CUBE_HIERARCHIES view, *6-155*
USER_CUBE_MEASURES view, *6-155*
USER_CUBE_NAMED_BUILD_SPECS view, *6-155*
USER_CUBE_SUB_PARTITION_LEVELS view, *6-155*
USER_CUBE_VIEW_COLUMNS view, *6-156*
USER_CUBE_VIEWS view, *6-156*
USER_CUBES view, *6-156*
USER_DATAPUMP_JOBS view, *6-156*
USER_DB_LINKS view, *6-156*
USER_DBFS_HS view, *6-157*
USER_DBFS_HS_COMMANDS view, *6-157*
USER_DBFS_HS_FILES view, *6-157*
USER_DBFS_HS_FIXED_PROPERTIES view, *6-157*
USER_DBFS_HS_PROPERTIES view, *6-158*
USER_DEPENDENCIES view, *6-158*

USER_DIM_ATTRIBUTES view, *6-158*
USER_DIM_CHILD_OF view, *6-158*
USER_DIM_HIERARCHIES view, *6-158*
USER_DIM_JOIN_KEY view, *6-158*
USER_DIM_LEVEL_KEY view, *6-159*
USER_DIM_LEVELS view, *6-159*
USER_DIMENSIONS view, *6-159*
USER_DUMP_DEST initialization parameter, *1-253*
USER_EDITIONED_TYPES view, *6-159*
USER_EDITIONING_VIEW_COLS view, *6-159*
USER_EDITIONING_VIEW_COLS_AE view, *6-159*
USER_EDITIONING_VIEWS view, *6-160*
USER_EDITIONING_VIEWS_AE view, *6-160*
USER_ENCRYPTED_COLUMNS view, *6-160*
USER_EPG_DAD_AUTHORIZATION view, *6-160*
USER_ERROR_TRANSLATIONS view, *6-160*
USER_ERRORS view, *6-160*
USER_ERRORS_AE view, *6-161*
USER_EVALUATION_CONTEXT_TABLES view, *6-161*
USER_EVALUATION_CONTEXT_VARS view, *6-161*
USER_EVALUATION_CONTEXTS view, *6-161*
USER_EXTENTS view, *6-161*
USER_EXTERNAL_LOCATIONS view, *6-162*
USER_EXTERNAL_TABLES view, *6-162*
USER_FILE_GROUP_EXPORT_INFO view, *6-162*
USER_FILE_GROUP_FILES view, *6-162*
USER_FILE_GROUP_TABLES view, *6-162*
USER_FILE_GROUP_TABLESPACES view, *6-162*
USER_FILE_GROUP_VERSIONS view, *6-163*
USER_FILE_GROUPS view, *6-163*
USER_FLASHBACK_ARCHIVE view, *6-163*
USER_FLASHBACK_ARCHIVE_TABLES view, *6-163*
USER_FLASHBACK_TXN_REPORT view, *6-163*
USER_FLASHBACK_TXN_STATE view, *6-164*
USER_FREE_SPACE view, *6-164*
USER_GOLDENGATE_PRIVILEGES view, *6-164*
USER_HEAT_MAP_SEG_HISTOGRAM view, *6-164*
USER_HEAT_MAP_SEGMENT view, *6-164*
USER_HISTOGRAMS synonym for USER_TAB_HISTOGRAMS, *6-165*
USER_HOST_ACES view, *6-165*
USER_IDENTIFIERS view, *6-165*
USER_ILMDATAMOVEMENTPOLICIES view, *6-165*
USER_ILMEVALUATIONDETAILS view, *6-166*
USER_ILMOBJECTS view, *6-166*
USER_ILMPOLICIES view, *6-166*
USER_ILMRESULTS view, *6-167*
USER_ILMTASKS view, *6-167*
USER_IND_COLUMNS view, *6-167*
USER_IND_EXPRESSIONS view, *6-167*
USER_IND_PARTITIONS view, *6-167*
USER_IND_PENDING_STATS view, *6-168*
USER_IND_STATISTICS view, *6-168*
USER_IND_SUBPARTITIONS view, *6-168*
USER_INDEXES view
  IND synonym, *6-102*
USER_INDEXES view *(continued)*
USER_INDEXTYPE_ARRAYTYPES view, *6-168*
USER_INDEXTYPE_COMMENTS view, *6-168*
USER_INDEXTYPE_OPERATORS view, *6-169*
USER_INDEXTYPES view, *6-169*
USER_INTERNAL_TRIGGERS view, *6-169*
USER_JAVA_ARGUMENTS view, *6-169*
USER_JAVA_CLASSES view, *6-169*
USER_JAVA_COMPILER_OPTIONS view, *6-170*
USER_JAVA_DERIVATIONS view, *6-170*
USER_JAVA_FIELDS view, *6-170*
USER_JAVA_IMPLEMENTS view, *6-170*
USER_JAVA_INNERS view, *6-170*
USER_JAVA_LAYOUTS view, *6-171*
USER_JAVA_METHODS view, *6-171*
USER_JAVA_NCOMPS view, *6-171*
USER_JAVA_POLICY view, *6-171*
USER_JAVA_RESOLVERS view, *6-171*
USER_JAVA_THROWS view, *6-172*
USER_JOBS view
  ALL_JOBS synonym, *2-138*
USER_JOIN_IND_COLUMNS view, *6-172*
USER_JSON_COLUMNS view, *6-172*
USER_LIBRARIES view, *6-173*
USER_LOB_PARTITIONS view, *6-173*
USER_LOB_SUBPARTITIONS view, *6-173*
USER_LOB_TEMPLATES view, *6-173*
USER_LOBS view, *6-173*
USER_LOG_GROUP_COLUMNS view, *6-173*
USER_LOG_GROUPS view, *6-174*
USER_MEASURE_FOLDER_CONTENTS view, *6-174*
USER_MEASURE_FOLDER_SUBFOLDERS view, *6-174*
USER_MEASURE_FOLDERS view, *6-174*
USER_METADATA_PROPERTIES view, *6-174*
USER_METHOD_PARAMS view, *6-175*
USER_METHOD_RESULTS view, *6-175*
USER_MINING_MODEL_ATTRIBUTES view, *6-175*
USER_MINING_MODEL_SETTINGS view, *6-175*
USER_MINING_MODELS view, *6-175*
USER_MP_GRAPH_WORKSPACES view, *2-5*
USER_MP_PARENT_WORKSPACES view, *2-5*
USER_MVIEW_AGGREGATES view, *6-176*
USER_MVIEW_ANALYSIS view, *6-176*
USER_MVIEW_COMMENTS view, *6-176*
USER_MVIEW_DETAIL_PARTITION view, *6-176*
USER_MVIEW_DETAIL_RELATIONS view, *6-176*
USER_MVIEW_DETAIL_SUBPARTITION view, *6-177*
USER_MVIEW_JOINS view, *6-177*
USER_MVIEW_KEYS view, *6-177*
USER_MVIEW_LOGS view, *6-177*
USER_MVIEW_REFRESH_TIMES view, *6-177*
USER_MVIEWS view, *6-177*
USER_NESTED_TABLE_COLS view, *6-177*
USER_NESTED_TABLES view, *6-178*

USER_NETWORK_ACL_PRIVILEGES view, *6-178*
USER_OBJ_AUDIT_OPTS view, *6-178*
USER_OBJ_COLATTRS view, *6-179*
USER_OBJECT_SIZE view, *6-179*
USER_OBJECT_TABLES view, *6-179*
USER_OBJECT_USAGE view, *6-179*
USER_OBJECTS view
  OBJ synonym, *6-106*
USER_OBJECTS_AE view, *6-180*
USER_OPANCILLARY view, *6-180*
USER_OPARGUMENTS view, *6-180*
USER_OPBINDINGS view, *6-180*
USER_OPERATOR_COMMENTS view, *6-181*
USER_OPERATORS view, *6-181*
USER_OUTLINE_HINTS view
  ALL_OUTLINE_HINTS synonym, *2-188*
USER_OUTLINES view
  ALL_OUTLINES synonym, *2-188*
USER_PARALLEL_EXECUTE_CHUNKS view, *6-181*
USER_PARALLEL_EXECUTE_TASKS view, *6-182*
USER_PART_COL_STATISTICS view, *6-182*
USER_PART_HISTOGRAMS view, *6-182*
USER_PART_INDEXES view, *6-182*
USER_PART_KEY_COLUMNS view, *6-182*
USER_PART_LOBS view, *6-183*
USER_PART_TABLES view, *6-183*
USER_PARTIAL_DROP_TABS view, *6-183*
USER_PASSWORD_LIMITS view, *6-183*
USER_PENDING_CONV_TABLES view, *6-183*
USER_PLSQL_COLL_TYPES view, *6-184*
USER_PLSQL_OBJECT_SETTINGS view, *6-184*
USER_PLSQL_TYPE_ATTRS view, *6-184*
USER_PLSQL_TYPES view, *6-184*
USER_POLICIES view, *6-184*
USER_POLICY_ATTRIBUTES view, *6-185*
USER_POLICY_CONTEXTS view, *6-185*
USER_POLICY_GROUPS view, *6-185*
USER_PRIVILEGE_MAP view, *6-185*
USER_PROCEDURES view, *6-186*
USER_PROXIES view, *6-186*
USER_QUEUE_SCHEDULES view, *6-186*
USER_QUEUE_SUBSCRIBERS view, *6-186*
USER_QUEUE_TABLES view, *6-186*
USER_QUEUES view, *6-187*
USER_RECYCLEBIN view
  RECYCLEBIN synonym, *6-114*
USER_REFRESH view, *6-187*
USER_REFRESH_CHILDREN view, *6-187*
USER_REFS view, *6-187*
USER_REGISTERED_MVIEWS view, *6-187*
USER_REGISTRY view, *6-187*
USER_REMOVED_WORKSPACES view, *2-5*
USER_REPCAT view, *2-3*
USER_REPCAT_TEMPLATE_OBJECTS view, *2-3*
USER_REPCAT_TEMPLATE_PARMS view, *2-3*
USER_REPCAT_TEMPLATE_SITES view, *2-3*
USER_REPCAT_USER_AUTHORIZATION view, *2-3*
USER_REPCAT_USER_PARM_VALUES view, *2-3*
USER_REPCATLOG view, *2-3*
USER_REPCOLUMN view, *2-4*
USER_REPCOLUMN_GROUP view, *2-4*
USER_REPCONFLICT view, *2-4*
USER_REPDDL view, *2-4*
USER_REPFLAVOR_COLUMNS view, *2-4*
USER_REPFLAVOR_OBJECTS view, *2-4*
USER_REPFLAVORS view, *2-4*
USER_REPGENERATED view, *2-4*
USER_REPGENOBJECTS view, *2-4*
USER_REPGROUP view, *2-4*
USER_REPGROUP_PRIVILEGES view, *2-4*
USER_REPGROUPED_COLUMN view, *2-4*
USER_REPKEY_COLUMNS view, *2-4*
USER_REPOBJECT view, *2-4*
USER_REPPARAMETER_COLUMN view, *2-4*
USER_REPPRIORITY view, *2-4*
USER_REPPRIORITY_GROUP view, *2-4*
USER_REPPROP view, *2-4*
USER_REPRESOL_STATS_CONTROL view, *2-4*
USER_REPRESOLUTION view, *2-4*
USER_REPRESOLUTION_METHOD view, *2-4*
USER_REPRESOLUTION_STATISTICS view, *2-4*
USER_REPSCHEMA view, *2-4*
USER_REPSITES view, *2-4*
USER_RESOURCE_LIMITS view, *6-188*
USER_RESUMABLE view, *6-188*
USER_REWRITE_EQUIVALENCES view, *6-188*
USER_ROLE_PRIVS view, *6-188*
USER_RSRC_CONSUMER_GROUP_PRIVS view, *6-189*
USER_RSRC_MANAGER_SYSTEM_PRIVS view, *6-189*
USER_RULE_SET_RULES view, *6-189*
USER_RULE_SETS view, *6-190*
USER_RULES view, *6-190*
USER_SCHEDULER_CHAIN_RULES view, *6-190*
USER_SCHEDULER_CHAIN_STEPS view, *6-190*
USER_SCHEDULER_CHAINS view, *6-190*
USER_SCHEDULER_CREDENTIALS view, *6-191*
USER_SCHEDULER_DB_DESTS view, *6-191*
USER_SCHEDULER_DESTS view, *6-191*
USER_SCHEDULER_FILE_WATCHERS view, *6-191*
USER_SCHEDULER_GROUP_MEMBERS view, *6-192*
USER_SCHEDULER_GROUPS view, *6-192*
USER_SCHEDULER_JOB_ARGS view, *6-192*
USER_SCHEDULER_JOB_DESTS view, *6-192*
USER_SCHEDULER_JOB_LOG view, *6-192*
USER_SCHEDULER_JOB_RUN_DETAILS view, *6-193*
USER_SCHEDULER_JOBS view, *6-193*
USER_SCHEDULER_NOTIFICATIONS view, *6-193*
USER_SCHEDULER_PROGRAM_ARGS view, *6-193*
USER_SCHEDULER_PROGRAMS view, *6-194*

USER_SCHEDULER_REMOTE_JOBSTATE view, 6-194
USER_SCHEDULER_RUNNING_CHAINS view, 6-194
USER_SCHEDULER_RUNNING_JOBS view, 6-194
USER_SCHEDULER_SCHEDULES view, 6-194
USER_SEC_RELEVANT_COLS view, 6-195
USER_SECONDARY_OBJECTS view, 6-195
USER_SEGMENTS view, 6-195
USER_SEQUENCES view
SEQ synonym, 6-121
USER_SOURCE view, 6-195
USER_SOURCE_AE view, 6-195
USER_SQL_TRANSLATION_PROFILES view, 6-196
USER_SQL_TRANSLATIONS view, 6-196
USER_SQLJ_TYPE_ATTRS view, 6-196
USER_SQLJ_TYPE_METHODS view, 6-196
USER_SQLJ_TYPES view, 6-196
USER_SQLSET view, 6-197
USER_SQLSET_BINDS view, 6-197
USER_SQLSET_PLANS view, 6-197
USER_SQLSET_REFERENCES view, 6-197
USER_SQLSET_STATEMENTS view, 6-197
USER_SQLTUNE_BINDS view, 6-197
USER_SQLTUNE_PLANS view, 6-197
USER_SQLTUNE_RATIONALE_PLAN view, 6-198
USER_SQLTUNE_STATISTICS view, 6-198
USER_SR_GRP_STATUS view, 6-198
USER_SR_GRP_STATUS_ALL view, 6-198
USER_SR_OBJ view, 6-198
USER_SR_OBJ_ALL view, 6-199
USER_SR_OBJ_STATUS view, 6-199
USER_SR_OBJ_STATUS_ALL view, 6-199
USER_SR_PARTN_OPS view, 6-199
USER_SR_STLOG_EXCEPTIONS view, 6-200
USER_SR_STLOG_STATS view, 6-200
USER_STAT_EXTENSIONS view, 6-200
USER_STORED_SETTINGS view, 6-200
USER_SUBPART_COL_STATISTICS view, 6-200
USER_SUBPART_HISTOGRAMS view, 6-201
USER_SUBPART_KEY_COLUMNS view, 6-201
USER_SUBPARTITION_TEMPLATES view, 6-201
USER_SUBSCR_REGISTRATIONS view, 6-201
USER_SYNONYMS view
SYN synonym, 6-123
USER_SYS_PRIVS view, 6-201
USER_TAB_COL_STATISTICS view, 6-202
USER_TAB_COLS view, 6-202
USER_TAB_COLUMNS view
COLS synonym, 3-187
USER_TAB_COMMENTS view, 6-203
USER_TAB_HISTGRM_PENDING_STATS view, 6-203
USER_TAB_HISTOGRAMS view
USER_HISTOGRAMS synonym, 6-165
USER_TAB_IDENTITY_COLS view, 6-203
USER_TAB_MODIFICATIONS view, 6-203
USER_TAB_PARTITIONS view, 6-204
USER_TAB_PENDING_STATS view, 6-204
USER_TAB_PRIVS view, 6-204
USER_TAB_PRIVS_MADE view, 6-204
USER_TAB_PRIVS_RECD view, 6-204
USER_TAB_STAT_PREFS view, 6-204
USER_TAB_STATISTICS view, 6-205
USER_TAB_STATS_HISTORY view, 6-205
USER_TAB_SUBPARTITIONS view, 6-205
USER_TABLES view
TABS synonym, 6-125
USER_TABLESPACES view, 6-205
USER_TRANSFORMATIONS view, 6-205
USER_TRIGGER_COLS view, 6-206
USER_TRIGGER_ORDERING view, 6-206
USER_TRIGGERS view, 6-206
USER_TS_QUOTAS view, 6-206
USER_TSTZ_TAB_COLS view, 6-206
USER_TSTZ_TABLES view, 6-207
USER_TUNE_MVIEW view, 6-207
USER_TYPE_ATTRS view, 6-207
USER_TYPE_METHODS view, 6-207
USER_TYPE_VERSIONS view, 6-207
USER_TYPES view, 6-208
USER_UNUSED_COL_TABS view, 6-208
USER_UPDATABLE_COLUMNS view, 6-208
USER_USERS view, 6-208
USER_USTATS view, 6-209
USER_VARRAYS view, 6-210
USER_VIEWS view, 6-210
USER_VIEWS_AE view, 6-210
USER_WALLET_ACES view, 6-210
USER_WARNING_SETTINGS view, 6-210
USER_WM_CONS_COLUMNS view, 2-5
USER_WM_CONSTRAINTS view, 2-5
USER_WM_IND_COLUMNS view, 2-5
USER_WM_IND_EXPRESSIONS view, 2-5
USER_WM_LOCKED_TABLES view, 2-5
USER_WM_MODIFIED_TABLES view, 2-5
USER_WM_PRIVS view, 2-5
USER_WM_RIC_INFO view, 2-5
USER_WM_TAB_TRIGGERS view, 2-5
USER_WM_VERSIONED_TABLES view, 2-5
USER_WM_VT_ERRORS view, 2-5
USER_WORKSPACE_PRIVS view, 2-5
USER_WORKSPACE_SAVEPOINTS view, 2-5
USER_WORKSPACES view, 2-6
USER_XML_INDEXES view, 6-211
USER_XML_NESTED_TABLES view, 6-211
USER_XML_OUT_OF_LINE_TABLES view, 6-211
USER_XML_SCHEMA_ATTRIBUTES view, 6-211
USER_XML_SCHEMA_COMPLEX_TYPES view, 6-211
USER_XML_SCHEMA_ELEMENTS view, 6-212
USER_XML_SCHEMA_NAMESPACES view, 6-212
USER_XML_SCHEMA_SIMPLE_TYPES view, 6-212

USER_XML_SCHEMA_SUBSTGRP_HEAD view, *6-212*
USER_XML_SCHEMA_SUBSTGRP_MBRS view, *6-213*
USER_XML_SCHEMAS view, *6-213*
USER_XML_TAB_COLS view, *6-213*
USER_XML_TABLES view, *6-213*
USER_XML_VIEW_COLS view, *6-213*
USER_XML_VIEWS view, *6-214*
USER_ZONEMAP_MEASURES view, *6-214*
USER_ZONEMAPS view, *6-214*
users
  authenticating, *1-176*
  multiple, *1-85*
UTL_FILE_DIR initialization parameter, *1-254*
UTLCHN1.SQL script, *B-3*
UTLDTREE.SQL script, *6-93*, *6-102*
UTLEXPT1.SQL script, *B-3*
UTLIRP.SQL script, *B-4*
UTLLOCKT.SQL script, *B-4*
UTLPWDMG.SQL script, *B-4*
UTLRP.SQL script, *B-4*
UTLXPLAN.SQL script, *B-5*

## V

V$ACCESS view, *7-3*
V$ACTIVE_INSTANCES view, *7-3*
V$ACTIVE_SERVICES view, *7-4*
V$ACTIVE_SESS_POOL_MTH view, *7-6*
V$ACTIVE_SESSION_HISTORY view, *7-6*
V$ADVISOR_PROGRESS view, *7-12*
V$ALERT_TYPES view, *7-13*
V$AQ view, *7-14*
V$AQ_BACKGROUND_COORDINATOR view, *7-14*
V$AQ_BMAP_NONDUR_SUBSCRIBERS view, *7-15*
V$AQ_CROSS_INSTANCE_JOBS view, *7-15*
V$AQ_JOB_COORDINATOR view, *7-16*
V$AQ_MESSAGE_CACHE view, *7-17*
V$AQ_NONDUR_REGISTRATIONS view, *7-18*
V$AQ_NONDUR_SUBSCRIBER view, *7-19*
V$AQ_NONDUR_SUBSCRIBER_LWM view, *7-20*
V$AQ_NOTIFICATION_CLIENTS view, *7-20*
V$AQ_SERVER_POOL view, *7-21*
V$AQ_SUBSCRIBER_LOAD view, *7-22*
V$ARCHIVE view, *7-23*
V$ARCHIVE_DEST view, *7-24*
V$ARCHIVE_DEST_STATUS view, *7-27*
V$ARCHIVE_GAP view, *7-29*
V$ARCHIVE_PROCESSES view, *7-30*
V$ARCHIVED_LOG view, *7-30*
V$ASM_ACFS_ENCRYPTION_INFO view, *7-33*
V$ASM_ACFS_SEC_ADMIN view, *7-34*
V$ASM_ACFS_SEC_CMDRULE view, *7-35*
V$ASM_ACFS_SEC_REALM view, *7-36*
V$ASM_ACFS_SEC_REALM_FILTER view, *7-37*
V$ASM_ACFS_SEC_REALM_GROUP view, *7-38*
V$ASM_ACFS_SEC_REALM_USER view, *7-39*
V$ASM_ACFS_SEC_RULE view, *7-40*
V$ASM_ACFS_SEC_RULESET view, *7-41*
V$ASM_ACFS_SEC_RULESET_RULE view, *7-42*
V$ASM_ACFS_SECURITY_INFO view, *7-43*
V$ASM_ACFSREPL view, *7-44*
V$ASM_ACFSREPLTAG view, *7-46*
V$ASM_ACFSSNAPSHOTS view, *7-47*
V$ASM_ACFSTAG view, *7-48*
V$ASM_ACFSVOLUMES view, *7-49*
V$ASM_ALIAS view, *7-50*
V$ASM_ATTRIBUTE view, *7-50*
V$ASM_AUDIT_CLEAN_EVENTS view, *7-51*
V$ASM_AUDIT_CLEANUP_JOBS view, *7-52*
V$ASM_AUDIT_CONFIG_PARAMS view, *7-53*
V$ASM_AUDIT_LAST_ARCH_TS view, *7-53*
V$ASM_CLIENT view, *7-54*
V$ASM_DISK view, *7-55*
V$ASM_DISK_IOSTAT view, *7-59*
V$ASM_DISK_STAT view, *7-60*
V$ASM_DISKGROUP view, *7-60*
V$ASM_DISKGROUP_STAT view, *7-62*
V$ASM_ESTIMATE view, *7-62*
V$ASM_FILE view, *7-63*
V$ASM_FILESYSTEM view, *7-65*
V$ASM_OPERATION view, *7-66*
V$ASM_TEMPLATE view, *7-67*
V$ASM_USER view, *7-68*
V$ASM_USERGROUP view, *7-69*
V$ASM_USERGROUP_MEMBER view, *7-69*
V$ASM_VOLUME view, *7-70*
V$ASM_VOLUME_STAT view, *7-71*
V$AW_AGGREGATE_OP view, *7-72*
V$AW_ALLOCATE_OP view, *7-72*
V$AW_CALC view, *7-73*
V$AW_LONGOPS view, *7-74*
V$AW_OLAP view, *7-75*
V$AW_SESSION_INFO view, *7-76*
V$BACKUP view, *7-77*
V$BACKUP_ARCHIVELOG_DETAILS view, *7-77*
V$BACKUP_ARCHIVELOG_SUMMARY view, *7-78*
V$BACKUP_ASYNC_IO view, *7-79*
V$BACKUP_CONTROLFILE_DETAILS view, *7-81*
V$BACKUP_CONTROLFILE_SUMMARY view, *7-81*
V$BACKUP_COPY_DETAILS view, *7-82*
V$BACKUP_COPY_SUMMARY view, *7-83*
V$BACKUP_CORRUPTION view, *7-84*
V$BACKUP_DATAFILE view, *7-85*
V$BACKUP_DATAFILE_DETAILS view, *7-87*
V$BACKUP_DATAFILE_SUMMARY view, *7-89*
V$BACKUP_DEVICE view, *7-89*
V$BACKUP_FILES view, *7-90*
V$BACKUP_NONLOGGED view, *7-93*
V$BACKUP_PIECE view, *7-94*

V$BACKUP_PIECE_DETAILS view, *7-95*
V$BACKUP_REDOLOG view, *7-97*
V$BACKUP_SET view, *7-98*
V$BACKUP_SET_DETAILS view, *7-100*
V$BACKUP_SET_SUMMARY view, *7-102*
V$BACKUP_SPFILE view, *7-103*
V$BACKUP_SPFILE_DETAILS view, *7-103*
V$BACKUP_SPFILE_SUMMARY view, *7-104*
V$BACKUP_SYNC_IO view, *7-104*
V$BGPROCESS view, *7-106*
V$BH view, *7-106*
V$BLOCK_CHANGE_TRACKING view, *7-108*
V$BLOCKING_QUIESCE view, *7-108*
V$BT_SCAN_CACHE view, *7-109*
V$BT_SCAN_OBJ_TEMPS view, *7-109*
V$BUFFER_POOL view, *7-110*
V$BUFFER_POOL_STATISTICS view, *7-111*
V$BUFFERED_PUBLISHERS view, *7-113*
V$BUFFERED_QUEUES view, *7-114*
V$BUFFERED_SUBSCRIBERS view, *7-115*
V$CACHE view, *7-116*
V$CACHE_LOCK view, *7-118*
V$CACHE_TRANSFER view, *7-119*
V$CIRCUIT view, *7-121*
V$CLASS_CACHE_TRANSFER view, *7-122*
V$CLIENT_SECRETS view, *7-123*
V$CLIENT_STATS view, *7-124*
V$CLONEDFILE view, *7-124*
V$CLUSTER_INTERCONNECTS view, *7-125*
V$CON_SYS_TIME_MODEL view, *7-126*
V$CON_SYSSTAT view, *7-126*
V$CON_SYSTEM_EVENT view, *7-127*
V$CON_SYSTEM_WAIT_CLASS view, *7-128*
V$CONFIGURED_INTERCONNECTS view, *7-129*
V$CONTAINERS view, *7-129*
V$CONTEXT view, *7-130*
V$CONTROLFILE view, *7-131*
V$CONTROLFILE_RECORD_SECTION view, *7-131*
V$COPY_CORRUPTION view, *7-132*
V$COPY_NONLOGGED view, *7-133*
V$CORRUPT_XID_LIST view, *7-134*
V$CPOOL_CC_INFO view, *7-134*
V$CPOOL_CC_STATS view, *7-135*
V$CPOOL_CONN_INFO view, *7-135*
V$CPOOL_STATS view, *7-137*
V$CR_BLOCK_SERVER view, *7-137*
V$CURRENT_BLOCK_SERVER view, *7-139*
V$DATABASE view, *7-140*
V$DATABASE_BLOCK_CORRUPTION view, *7-147*
V$DATABASE_INCARNATION view, *7-148*
V$DATAFILE view, *7-149*
V$DATAFILE_COPY view, *7-151*
V$DATAFILE_HEADER view, *7-154*
V$DATAGUARD_CONFIG view, *7-155*
V$DATAGUARD_STATS view, *7-156*
V$DATAGUARD_STATUS view, *7-157*
V$DB_CACHE_ADVICE view, *7-158*
V$DB_OBJECT_CACHE view, *7-159*
V$DB_PIPES view, *7-161*
V$DB_TRANSPORTABLE_PLATFORM view, *7-161*
V$DBFILE view, *7-162*
V$DBLINK view, *7-162*
V$DEAD_CLEANUP view, *7-163*
V$DELETED_OBJECT view, *7-164*
V$DG_BROKER_CONFIG view, *7-165*
V$DIAG_INFO view, *7-166*
V$DISPATCHER view, *7-166*
V$DISPATCHER_CONFIG view, *7-167*
V$DISPATCHER_RATE view, *7-168*
V$DNFS_CHANNELS view, *7-173*
V$DNFS_FILES view, *7-174*
V$DNFS_SERVERS view, *7-175*
V$DNFS_STATS view, *7-175*
V$DYNAMIC_REMASTER_STATS view, *7-177*
V$EDITIONABLE_TYPES view, *7-178*
V$EMON view, *7-178*
V$EMX_USAGE_STATS view, *7-180*
V$ENABLEDPRIVS view, *7-181*
V$ENCRYPTED_TABLESPACES view, *7-181*
V$ENCRYPTION_KEYS view, *7-182*
V$ENCRYPTION_WALLET view, *7-184*
V$ENQUEUE_LOCK view, *7-185*
V$ENQUEUE_STAT view, *7-186*
V$ENQUEUE_STATISTICS view, *7-186*
V$EVENT_HISTOGRAM view, *7-187*
V$EVENT_HISTOGRAM_MICRO view, *7-188*
V$EVENT_NAME view, *7-189*
V$EVENTMETRIC view, *7-189*
V$EXECUTION view, *7-190*
V$FALSE_PING view, *7-191*
V$FAST_START_SERVERS view, *7-192*
V$FAST_START_TRANSACTIONS view, *7-192*
V$FILE_CACHE_TRANSFER view, *7-193*
V$FILE_HISTOGRAM view, *7-194*
V$FILEMETRIC view, *7-195*
V$FILEMETRIC_HISTORY view, *7-195*
V$FILESPACE_USAGE view, *7-196*
V$FILESTAT view, *7-196*
V$FIXED_TABLE view, *7-197*
V$FIXED_VIEW_DEFINITION view, *7-198*
V$FLASHBACK_DATABASE_LOG view, *7-198*
V$FLASHBACK_DATABASE_LOGFILE view, *7-199*
V$FLASHBACK_DATABASE_STAT view, *7-200*
V$FLASHBACK_TXN_GRAPH view, *7-200*
V$FLASHBACK_TXN_MODS view, *7-201*
V$FLASHFILESTAT view, *7-203*
V$FOREIGN_ARCHIVED_LOG view, *7-203*
V$FS_FAILOVER_STATS view, *7-206*
V$FS_OBSERVER_HISTOGRAM view, *7-206*
V$GC_ELEMENT view, *7-207*

V$GC_ELEMENTS_WITH_COLLISIONS view, *7-208*
V$GCSHVMASTER_INFO view, *7-208*
V$GCSPFMASTER_INFO view, *7-209*
V$GES_BLOCKING_ENQUEUE view, *7-209*
V$GES_CONVERT_LOCAL view, *7-211*
V$GES_CONVERT_REMOTE view, *7-211*
V$GES_ENQUEUE view, *7-212*
V$GES_LATCH view, *7-214*
V$GES_RESOURCE view, *7-214*
V$GES_STATISTICS view, *7-214*
V$GG_APPLY_COORDINATOR view, *7-215*
V$GG_APPLY_READER view, *7-217*
V$GG_APPLY_RECEIVER view, *7-219*
V$GG_APPLY_SERVER view, *7-220*
V$GLOBAL_BLOCKED_LOCKS view, *7-223*
V$GLOBAL_TRANSACTION view, *7-224*
V$GOLDENGATE_CAPTURE view, *7-225*
V$GOLDENGATE_MESSAGE_TRACKING view, *7-228*
V$GOLDENGATE_TABLE_STATS view, *7-230*
V$GOLDENGATE_TRANSACTION view, *7-231*
V$HANG_INFO view, *7-233*
V$HANG_SESSION_INFO view, *7-235*
V$HANG_STATISTICS view, *7-236*
V$HEAT_MAP_SEGMENT view, *7-236*
V$HM_CHECK view, *7-237*
V$HM_CHECK_PARAM view, *7-238*
V$HM_FINDING view, *7-238*
V$HM_INFO view, *7-239*
V$HM_RECOMMENDATION view, *7-240*
V$HM_RUN view, *7-241*
V$HS_AGENT view, *7-242*
V$HS_PARAMETER view, *7-242*
V$HS_SESSION view, *7-243*
V$HVMASTER_INFO view, *7-243*
V$IM_COLUMN_LEVEL view, *8-1*
V$IM_SEGMENTS view, *8-2*
V$IM_USER_SEGMENTS view, *8-3*
V$INDEXED_FIXED_COLUMN view, *8-5*
V$INMEMORY_AREA view, *8-6*
V$INSTANCE view, *8-6*
V$INSTANCE_CACHE_TRANSFER view, *8-8*
V$INSTANCE_PING view, *8-10*
V$INSTANCE_RECOVERY view, *8-11*
V$IO_CALIBRATION_STATUS view, *8-13*
V$IO_OUTLIER view, *8-13*
V$IOFUNCMETRIC view, *8-14*
V$IOFUNCMETRIC_HISTORY view, *8-15*
V$IOSTAT_CONSUMER_GROUP view, *8-15*
V$IOSTAT_FILE view, *8-16*
V$IOSTAT_FUNCTION view, *8-17*
V$IOSTAT_FUNCTION_DETAIL view, *8-18*
V$IOSTAT_NETWORK view, *8-19*
V$JAVA_LIBRARY_CACHE_MEMORY view, *8-20*
V$JAVA_POOL_ADVICE view, *8-21*
V$KERNEL_IO_OUTLIER view, *8-21*
V$KEY_VECTOR view, *8-22*
V$LATCH view, *8-25*
V$LATCH_CHILDREN view, *8-26*
V$LATCH_MISSES view, *8-27*
V$LATCH_PARENT view, *8-28*
V$LATCHHOLDER view, *8-28*
V$LATCHNAME view, *8-28*
V$LGWRIO_OUTLIER view, *8-29*
V$LIBCACHE_LOCKS view, *8-30*
V$LIBRARY_CACHE_MEMORY view, *8-31*
V$LIBRARYCACHE view, *8-31*
V$LICENSE view, *8-32*
V$LOADISTAT view, *8-33*
V$LOADPSTAT view, *8-33*
V$LOCK view, *8-34*
V$LOCK_ACTIVITY view, *8-36*
V$LOCK_TYPE view, *8-37*
V$LOCKED_OBJECT view, *8-37*
V$LOG view, *8-38*
V$LOG_HISTORY view, *8-39*
V$LOGFILE view, *8-40*
V$LOGHIST view, *8-40*
V$LOGMNR_CONTENTS view, *8-41*
V$LOGMNR_DICTIONARY view, *8-47*
V$LOGMNR_DICTIONARY_LOAD view, *8-48*
V$LOGMNR_LATCH view, *8-48*
V$LOGMNR_LOGS view, *8-49*
V$LOGMNR_PARAMETERS view, *8-50*
V$LOGMNR_PROCESS view, *8-51*
V$LOGMNR_SESSION view, *8-52*
V$LOGMNR_STATS view, *8-54*
V$LOGSTDBY view, *8-55*
V$LOGSTDBY_PROCESS view, *8-56*
V$LOGSTDBY_PROGRESS view, *8-57*
V$LOGSTDBY_STATE view, *8-58*
V$LOGSTDBY_STATS view, *8-59*
V$LOGSTDBY_TRANSACTION view, *8-60*
V$MANAGED_STANDBY view, *8-62*
V$MAP_COMP_LIST view, *8-63*
V$MAP_ELEMENT view, *8-64*
V$MAP_EXT_ELEMENT view, *8-65*
V$MAP_FILE view, *8-65*
V$MAP_FILE_EXTENT view, *8-66*
V$MAP_FILE_IO_STACK view, *8-67*
V$MAP_LIBRARY view, *8-68*
V$MAP_SUBELEMENT view, *8-68*
V$MAPPED_SQL view, *8-69*
V$MEMORY_CURRENT_RESIZE_OPS view, *8-70*
V$MEMORY_DYNAMIC_COMPONENTS view, *8-70*
V$MEMORY_RESIZE_OPS view, *8-71*
V$MEMORY_TARGET_ADVICE view, *8-72*
V$METRIC view, *8-73*
V$METRIC_HISTORY view, *8-75*
V$METRICGROUP view, *8-74*
V$METRICNAME view, *8-76*

V$MTTR_TARGET_ADVICE view, *8-76*
V$MUTEX_SLEEP view, *8-77*
V$MUTEX_SLEEP_HISTORY view, *8-78*
V$MVREFRESH view, *8-79*
V$MYSTAT view, *8-79*
V$NFS_CLIENTS view, *8-79*
V$NFS_LOCKS view, *8-80*
V$NFS_OPEN_FILES view, *8-81*
V$NLS_PARAMETERS view, *8-81*
V$NLS_VALID_VALUES view, *8-82*
V$NONLOGGED_BLOCK view, *8-82*
V$OBJECT_DEPENDENCY view, *8-83*
V$OBJECT_PRIVILEGE view, *8-84*
V$OBJECT_USAGE view, *8-84*
V$OBSOLETE_BACKUP_FILES view, *8-85*
V$OBSOLETE_PARAMETER view, *8-87*
V$OFFLINE_RANGE view, *8-88*
V$OPEN_CURSOR view, *8-88*
V$OPTIMIZER_PROCESSING_RATE view, *8-89*
V$OPTION view, *8-90*
V$OSSTAT view, *8-91*
V$PARALLEL_DEGREE_LIMIT_MTH view, *8-93*
V$PARAMETER view, *8-93*
V$PARAMETER_VALID_VALUES view, *8-95*
V$PARAMETER2 view, *8-95*
V$PATCHES view, *8-97*
V$PDB_INCARNATION view, *8-98*
V$PDBS view, *8-99*
V$PERSISTENT_PUBLISHERS view, *8-100*
V$PERSISTENT_QMN_CACHE view, *8-100*
V$PERSISTENT_QUEUES view, *8-102*
V$PERSISTENT_SUBSCRIBERS view, *8-103*
V$PGA_TARGET_ADVICE view, *8-105*
V$PGA_TARGET_ADVICE_HISTOGRAM view, *8-106*
V$PGASTAT view, *8-108*
V$PQ_SESSTAT view, *8-110*
V$PQ_SLAVE view, *8-111*
V$PQ_SYSSTAT view, *8-112*
V$PQ_TQSTAT view, *8-112*
V$PROCESS view, *8-113*
V$PROCESS_MEMORY view, *8-115*
V$PROPAGATION_RECEIVER view, *8-115*
V$PROPAGATION_SENDER view, *8-117*
V$PROXY_ARCHIVEDLOG view, *8-118*
V$PROXY_ARCHIVELOG_DETAILS view, *8-120*
V$PROXY_ARCHIVELOG_SUMMARY view, *8-121*
V$PROXY_COPY_DETAILS view, *8-122*
V$PROXY_COPY_SUMMARY view, *8-123*
V$PROXY_DATAFILE view, *8-124*
V$PWFILE_USERS view, *8-127*
V$PX_INSTANCE_GROUP view, *8-127*
V$PX_PROCESS view, *8-128*
V$PX_PROCESS_SYSSTAT view, *8-128*
V$PX_SESSION view, *8-130*
V$PX_SESSTAT view, *8-130*
V$QMON_COORDINATOR_STATS view, *8-131*
V$QMON_SERVER_STATS view, *8-132*
V$QMON_TASK_STATS view, *8-133*
V$QMON_TASKS view, *8-134*
V$QUEUE view, *8-135*
V$QUEUEING_MTH view, *8-135*
V$RECOVER_FILE view, *8-135*
V$RECOVERY_AREA_USAGE view, *8-136*
V$RECOVERY_FILE_DEST view, *8-137*
V$RECOVERY_FILE_STATUS view, *8-137*
V$RECOVERY_LOG view, *8-138*
V$RECOVERY_PROGRESS view, *8-139*
V$RECOVERY_STATUS view, *8-140*
V$REDO_DEST_RESP_HISTOGRAM view, *8-141*
V$REPLPROP view, *8-141*
V$REPLQUEUE view, *8-143*
V$REQDIST view, *8-143*
V$RESERVED_WORDS view, *8-144*
V$RESOURCE view, *8-144*
V$RESOURCE_LIMIT view, *8-145*
V$RESTORE_POINT view, *8-146*
V$RESULT_CACHE_DEPENDENCY view, *8-147*
V$RESULT_CACHE_MEMORY view, *8-147*
V$RESULT_CACHE_OBJECTS view, *8-148*
V$RESULT_CACHE_STATISTICS view, *8-149*
V$RMAN_BACKUP_JOB_DETAILS view, *8-150*
V$RMAN_BACKUP_SUBJOB_DETAILS view, *8-152*
V$RMAN_BACKUP_TYPE view, *8-153*
V$RMAN_COMPRESSION_ALGORITHM view, *8-154*
V$RMAN_CONFIGURATION view, *8-155*
V$RMAN_ENCRYPTION_ALGORITHMS view, *8-155*
V$RMAN_OUTPUT view, *8-156*
V$RMAN_STATUS view, *8-156*
V$RO_USER_ACCOUNT view, *8-158*
V$ROLLNAME view, *8-159*
V$ROLLSTAT view, *8-159*
V$ROWCACHE view, *8-160*
V$ROWCACHE_PARENT view, *8-161*
V$ROWCACHE_SUBORDINATE view, *8-162*
V$RSRC_CONS_GROUP_HISTORY view, *8-163*
V$RSRC_CONSUMER_GROUP view, *8-166*
V$RSRC_CONSUMER_GROUP_CPU_MTH view, *8-169*
V$RSRC_PLAN view, *8-170*
V$RSRC_PLAN_CPU_MTH view, *8-171*
V$RSRC_PLAN_HISTORY view, *8-172*
V$RSRC_SESSION_INFO view, *8-173*
V$RSRCMGRMETRIC view, *8-178*
V$RSRCMGRMETRIC_HISTORY view, *8-180*
V$RULE view, *8-181*
V$RULE_SET view, *8-181*
V$RULE_SET_AGGREGATE_STATS view, *8-183*
V$SCHEDULER_RUNNING_JOBS view, *9-1*
V$SECUREFILE_TIMER view, *9-1*

V$SEGMENT_STATISTICS view, 9-2
V$SEGSTAT view, 9-3
V$SEGSTAT_NAME view, 9-3
V$SERV_MOD_ACT_STATS view, 9-4
V$SERVICE_EVENT view, 9-4
V$SERVICE_STATS view, 9-5
V$SERVICE_WAIT_CLASS view, 9-6
V$SERVICEMETRIC view, 9-6
V$SERVICEMETRIC_HISTORY view, 9-7
V$SERVICES view, 9-8
V$SES_OPTIMIZER_ENV view, 9-10
V$SESS_IO view, 9-10
V$SESS_TIME_MODEL view, 9-11
V$SESSION view, 9-14
V$SESSION_BLOCKERS view, 9-20
V$SESSION_CONNECT_INFO view, 9-21
V$SESSION_CURSOR_CACHE view, 9-22
V$SESSION_EVENT view, 9-23, C-1
V$SESSION_FIX_CONTROL view, 9-24
V$SESSION_LONGOPS view, 9-25
V$SESSION_OBJECT_CACHE view, 9-27
V$SESSION_WAIT view, 9-27, C-1
V$SESSION_WAIT_CLASS view, 9-29
V$SESSION_WAIT_HISTORY view, 9-30
V$SESSIONS_COUNT view, 9-30
V$SESSMETRIC view, 9-31
V$SESSTAT view, 9-31, E-1
V$SGA view, 9-32
V$SGA_CURRENT_RESIZE_OPS view, 9-32
V$SGA_DYNAMIC_COMPONENTS view, 9-33
V$SGA_DYNAMIC_FREE_MEMORY view, 9-34
V$SGA_RESIZE_OPS view, 9-34
V$SGA_TARGET_ADVICE view, 9-35
V$SGAINFO view, 9-36
V$SGASTAT view, 9-36
V$SHARED_POOL_ADVICE view, 9-37
V$SHARED_POOL_RESERVED view, 9-38
V$SHARED_SERVER view, 9-39
V$SHARED_SERVER_MONITOR view, 9-40
V$SORT_SEGMENT view, 9-40
V$SPPARAMETER view, 9-41
V$SQL view, 9-42
V$SQL_BIND_CAPTURE view, 9-47
V$SQL_BIND_DATA view, 9-49
V$SQL_BIND_METADATA view, 9-50
V$SQL_CS_HISTOGRAM view, 9-51
V$SQL_CS_SELECTIVITY view, 9-51
V$SQL_CS_STATISTICS view, 9-52
V$SQL_CURSOR view, 9-53
V$SQL_JOIN_FILTER view, 9-54
V$SQL_MONITOR view, 9-54
V$SQL_MONITOR_SESSTAT view, 9-59
V$SQL_MONITOR_STATNAME view, 9-60
V$SQL_OPTIMIZER_ENV view, 9-60
V$SQL_PLAN view, 9-61
V$SQL_PLAN_MONITOR view, 9-64
V$SQL_PLAN_STATISTICS view, 9-69
V$SQL_PLAN_STATISTICS_ALL view, 9-70
V$SQL_REDIRECTION view, 9-75
V$SQL_SHARED_CURSOR view, 9-76
V$SQL_SHARED_MEMORY view, 9-79
V$SQL_WORKAREA view, 9-80
V$SQL_WORKAREA_ACTIVE view, 9-82
V$SQL_WORKAREA_HISTOGRAM view, 9-84
V$SQLAREA view, 9-85
V$SQLAREA_PLAN_HASH view, 9-90
V$SQLCOMMAND view, 9-94
V$SQLFN_ARG_METADATA view, 9-95
V$SQLFN_METADATA view, 9-95
V$SQLSTATS view, 9-96
V$SQLSTATS_PLAN_HASH view, 9-101
V$SQLTEXT view, 9-101
V$SQLTEXT_WITH_NEWLINES view, 9-101
V$STANDBY_EVENT_HISTOGRAM view, 9-102
V$STANDBY_LOG view, 9-103
V$STATISTICS_LEVEL view, 9-104
V$STATNAME view, 9-105, E-1
V$STREAMS_APPLY_COORDINATOR view, 9-106
V$STREAMS_APPLY_READER view, 9-108
V$STREAMS_APPLY_SERVER view, 9-111
V$STREAMS_CAPTURE view, 9-114
V$STREAMS_MESSAGE_TRACKING view, 9-117
V$STREAMS_POOL_ADVICE view, 9-118
V$STREAMS_POOL_STATISTICS view, 9-119
V$STREAMS_TRANSACTION view, 9-120
V$SUBCACHE view, 9-121
V$SUBSCR_REGISTRATION_STATS view, 9-122
V$SYS_OPTIMIZER_ENV view, 9-123
V$SYS_TIME_MODEL view, 9-124
V$SYSAUX_OCCUPANTS view, 9-124
V$SYSMETRIC view, 9-125
V$SYSMETRIC_HISTORY view, 9-125
V$SYSMETRIC_SUMMARY view, 9-126
V$SYSSTAT view, 9-127, E-1
V$SYSTEM_CURSOR_CACHE view, 9-128
V$SYSTEM_EVENT view, 9-128, C-1
V$SYSTEM_FIX_CONTROL view, 9-129
V$SYSTEM_PARAMETER view, 9-130
V$SYSTEM_PARAMETER2 view, 9-131
V$SYSTEM_WAIT_CLASS view, 9-133
V$TABLESPACE view, 9-134
V$TEMP_CACHE_TRANSFER view, 9-134
V$TEMP_EXTENT_MAP view, 9-135
V$TEMP_EXTENT_POOL view, 9-136
V$TEMP_SPACE_HEADER view, 9-137
V$TEMPFILE view, 9-137
V$TEMPORARY_LOBS view, 9-138
V$TEMPSEG_USAGE view, 9-138
V$TEMPSTAT view, 9-139
V$TEMPUNDOSTAT view, 9-140

V$THREAD view, *9-141*
V$THRESHOLD_TYPES view, *9-142*
V$TIMER view, *9-143*
V$TIMEZONE_NAMES view, *9-143*
V$TOPLEVELCALL view, *9-143*
V$TRANSACTION view, *9-144*
V$TRANSACTION_ENQUEUE view, *9-146*
V$TRANSPORTABLE_PLATFORM view, *9-146*
V$TSDP_SUPPORTED_FEATURE view, *9-147*
V$TYPE_SIZE view, *9-147*
V$UNDOSTAT view, *9-148*
V$UNUSABLE_BACKUPFILE_DETAILS view, *9-150*
V$VERSION view, *9-151*
V$VPD_POLICY view, *9-151*
V$WAIT_CHAINS view, *9-152*
V$WAITCLASSMETRIC view, *9-154*
V$WAITCLASSMETRIC_HISTORY view, *9-155*
V$WAITSTAT view, *9-155*
V$WALLET view, *9-155*
V$WORKLOAD_REPLAY_THREAD view, *9-156*
V$XML_AUDIT_TRAIL view, *9-158*
V$XSTREAM_APPLY_COORDINATOR view, *9-160*
V$XSTREAM_APPLY_READER view, *9-163*
V$XSTREAM_APPLY_RECEIVER view, *9-165*
V$XSTREAM_APPLY_SERVER view, *9-166*
V$XSTREAM_CAPTURE view, *9-169*
V$XSTREAM_MESSAGE_TRACKING view, *9-173*
V$XSTREAM_OUTBOUND_SERVER view, *9-174*
V$XSTREAM_TABLE_STATS view, *9-177*
V$XSTREAM_TRANSACTION view, *9-179*
views
  data dictionary views, *2-1*

## W

wait events
  alter system set dispatcher, *C-5*
  ARCH Remote Write, *C-5*
  ASYNC Remote Write, *C-5*
  batched allocate scn lock request, *C-5*
  BFILE check if exists, *C-6*
  BFILE check if open, *C-6*
  BFILE closure, *C-6*
  BFILE get length, *C-6*
  BFILE get name object, *C-6*
  BFILE get path object, *C-7*
  BFILE internal seek, *C-7*
  BFILE open, *C-7*
  BFILE read, *C-7*
  broadcast mesg queue transition, *C-7*
  broadcast mesg recovery queue transition, *C-8*
  buffer busy waits, *C-8*
  buffer deadlock, *C-9*
  buffer latch, *C-9*
  buffer read retry, *C-10*
  checkpoint completed, *C-10*
wait events *(continued)*
  cleanup of aborted processes, *C-10*
  controlfile parallel write, *C-10*
  controlfile sequential read, *C-11*
  controlfile single write, *C-11*
  cursor: mutex S, *C-11*
  cursor: mutex X, *C-12*
  cursor: pin S, *C-12*
  cursor: pin S wait on X, *C-12*
  cursor: pin X, *C-13*
  Data Guard Broker: single instance, *C-13*
  Data Guard: process cleanup, *C-13*
  Data Guard: process exit, *C-13*
  db file asynch I/O submit, *C-13*
  db file parallel read, *C-13*
  db file parallel write, *C-14*
  db file scattered read, *C-14*
  db file sequential read, *C-15*
  db file single write, *C-15*
  DFS db file lock, *C-16*
  DFS lock handle, *C-16*
  direct path read, *C-16*
  direct path sync, *C-17*
  direct path write, *C-17*
  Disk file operations I/O, *C-18*
  dispatcher shutdown, *C-18*
  dispatcher timer, *C-18*
  duplicate cluster key, *C-19*
  enq: OW - initialization, *C-19*
  enq: OW - termination, *C-19*
  enqueue, *C-19*
  flashback buf free by RVWR, *C-20*
  flashback logfile sync, *C-20*
  free buffer waits, *C-20*
  free global transaction table entry, *C-21*
  free process state object, *C-21*
  gc recovery quiesce, *C-21*
  GCS lock open S, *C-21*
  GCS lock open X, *C-22*
  gcs remastering wait for drop pkey, *C-22*
  global cache busy, *C-22*
  global cache freelist, *C-23*
  global cache lock cleanup, *C-22*
  inactive session, *C-23*
  inactive transaction branch, *C-23*
  index block split, *C-23*
  instance state change, *C-24*
  io done, *C-24*
  kcl bg acks, *C-24*
  ksxr wait for mount shared, *C-25*
  ktm: instance recovery, *C-25*
  latch activity, *C-25*
  latch free, *C-26*
  library cache load lock, *C-26*
  library cache lock, *C-26*
  library cache pin, *C-27*

wait events *(continued)*
LMON global data update, C-27
lock manager wait for remote message, C-28
Log archive I/O, C-28
log buffer space, C-28
log file parallel write, C-28
log file sequential read, C-29
log file single write, C-29
log file switch (archiving needed), C-29
log file switch (checkpoint incomplete), C-29
log file switch (clearing log file), C-30
log file switch (private strand flush incomplete), C-30
log file switch completion, C-30
log file sync, C-30
log switch/archive, C-30
optimizer stats update retry, C-31
parallel recovery change buffer free, C-31
parallel recovery control message reply, C-31
parallel recovery coord send blocked, C-31
parallel recovery coord wait for reply, C-31
parallel recovery coordinator waits for slave cleanup, C-31
parallel recovery read buffer free, C-32
parallel recovery slave next change, C-32
pending global transaction(s), C-32
pipe get, C-32
pipe put, C-32
PL/SQL lock timer, C-33
pmon timer, C-33
prewarm transfer retry, C-33
prior process spawner to be cleaned up, C-33
process startup, C-34
PX dequeue wait, C-34
PX qref latch, C-34
PX server shutdown, C-34
PX signal server, C-35
rdbms ipc message, C-35
rdbms ipc message block, C-35
rdbms ipc reply, C-35
read by other session, C-36
recovery read, C-36
Redo Transport Attach, C-36
Redo Transport Close, C-36
Redo Transport Detach, C-37
Redo Transport Open, C-37
Redo Transport Ping, C-37
Redo Transport Slave Shutdown, C-37
Redo Transport Slave Startup, C-37
Redo Writer Remote Sync Complete, C-37
Redo Writer Remote Sync Notify, C-37
Remote Sync Ping, C-38
resmgr:become active, C-38
resmgr:cpu quantum, C-38
resmgr:pq queued, C-38
rolling migration: cluster quiesce, C-38
wait events *(continued)*
row cache lock, C-39
RVWR wait for flashback copy, C-39
sbtbufinfo, C-39
sbtgetbuf, C-39
sbtmapbuf, C-40
sbtrelbuf, C-40
scginq AST call, C-40
SGA: allocation forcing component growth, C-40
SGA: MMAN sleep for component shrink, C-40
SGA: sga_target resize, C-40
Shared IO Pool Memory, C-41
shared server idle event, C-41
single-task message, C-41
smon timer, C-41
SQL*Net break/reset to client, C-41
SQL*Net break/reset to dblink, C-42
SQL*Net message from client, C-42
SQL*Net message from dblink, C-42
SQL*Net message to client, C-43
SQL*Net message to dblink, C-43
SQL*Net more data from client, C-43
SQL*Net more data from dblink, C-43
SQL*Net more data to client, C-44
SQL*Net more data to dblink, C-44
Streams AQ: waiting for messages in the queue, C-44
switch logfile command, C-44
SYNC Remote Write, C-45
timer in sksawat, C-45
transaction, C-45
unbound tx, C-46
undo segment extension, C-46
undo segment recovery, C-46
undo segment tx slot, C-46
undo_retention publish retry, C-46
utl_file I/O, C-47
virtual circuit status, C-47
virtual circuit wait, C-47
WCR: replay client notify, C-47
WCR: replay clock, C-48
WCR: replay lock order, C-48
WCR: replay paused, C-48
WCR: Sync context busy, C-48
WMON goes to sleep, C-48
write complete waits, C-48
writes stopped by instance recovery or database suspension, C-49
WM_COMPRESS_BATCH_SIZES view, 2-6
WM_COMPRESSIBLE_TABLES view, 2-6
WM_EVENTS_INFO view, 2-6
WM_INSTALLATION view, 2-6
WM_REPLICATION_INFO view, 2-6
WORKAREA_SIZE_POLICY initialization parameter, 1-255

## X

XML_DB_EVENTS initialization parameter, *1-255*