Oracle® Database

2 Day DBA

12*c* Release 1 (12.1)

E51671-08

May 2015

ORACLE®

Oracle Database 2 Day DBA, 12c Release 1 (12.1)

E51671-08

Primary Author: Bert Rich

Contributing Authors: Kathy Rich, Janet Stern, Steve Fogel, Colin McGregor, Lance Ashdown, Eric Belden, Craig Foch, Sushil Kumar, Roza Leyderman, Paul Needham, and Douglas Williams

Contributors: Bharat Baddepudi, Prasad Bagal, Timothy Chien, Benoit Dageville, Sudip Datta, Kakali Das, Mark Dilman, Prabhaker Gongloor, Cecilia Grant, Shivani Gupta, Pat Huey, Chaitanya Koratamaddi, Balaji Krishnan, Vasudha Krishnaswamy, Rich Long, Venkat Maddali, Matthew McKerley, Ed Miner, Mughees Minhas, Saurabh Pandey, Kant Patel, Malai Stalin, Mark Townsend, Xiaofang Wang, Kat Weill, Khaled Yagoub, and Mike Zampiceni

Contents

3 Getting Started with Database Administration

4 Configuring the Network Environment

5 Managing the Oracle Instance

6 Managing Database Storage Structures

7 Administering User Accounts and Security

8 Managing Schema Objects

9 Performing Backup and Recovery

10 Monitoring and Tuning the Database

11 Managing PDBs with EM Express

Index

Preface

Oracle Database 2 Day DBA is a database administration quick start guide that teaches you how to perform day-to-day database administrative tasks. The goal of this book is to help you understand the concepts behind Oracle Database. It teaches you how to perform all common administration tasks needed to keep the database operational, including how to perform basic troubleshooting and performance monitoring activities.

Audience

Oracle Database 2 Day DBA is for anyone who wants to perform basic administrative tasks with Oracle Database. Only minimal basic knowledge of or experience with database management is required, and a basic knowledge of computers.

In particular, this guide is for the following groups of Oracle users:

- Developers who want to acquire basic database administrator (DBA) skills
- Anyone managing database servers
- Database administrators managing an Oracle database for a small or medium-sized business

This book is equally useful for enterprise DBAs. It recommends best practices and describes efficient ways of performing administrative tasks with Oracle Enterprise Manager as the primary interface.

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 Documentation

For more information, see these Oracle resources:

- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database SQL Language Reference*
- *Oracle Database Reference*
- *Oracle Database Backup and Recovery User's Guide*
- *Oracle Real Application Clusters Installation Guide for Linux and UNIX* or other operating system
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle Clusterware Administration and Deployment Guide*
- *Oracle Automatic Storage Management Administrator's Guide*
- *Oracle Database Installation Guide*

There are also several other 2 Day + books available for a variety of topics:

- *Oracle Database 2 Day + Security Guide*
- *Oracle Database 2 Day + Performance Tuning Guide*
- *Oracle Database 2 Day + Real Application Clusters Guide*
- *Oracle Database 2 Day Developer's Guide*
- *Oracle Database 2 Day + .NET Developer's Guide for Microsoft Windows*
- *Oracle Database 2 Day + PHP Developer's Guide*
- *Oracle Database 2 Day + Application Express Developer's Guide*
- *Oracle Database 2 Day + Java Developer's Guide*

Many of the examples in this guide use the sample schemas, which are installed by default when you select the Basic Installation option with an Oracle Database installation. See *Oracle Database Sample Schemas* for information about how these schemas were created and how you can use them.

To download free release notes, installation documentation, updated Oracle documentation, white papers, or other collateral, 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`

If you have a user name and password for OTN, then 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 2 Day DBA

This preface lists changes in *Oracle Database 2 Day DBA*.

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

The following are changes in *Oracle Database 2 Day DBA* for Oracle Database 12*c* Release 1 (12.1.0.2).

New Features

Oracle Enterprise Manager Database Express (EM Express) can be used to manage multitenant container databases (CDBs) and pluggable databases (PDBs), in addition to managing non-CDBs:

- See "Starting EM Express for a CDB" for information about starting EM Express to manage a CDB.
- See "Starting EM Express for a PDB" for information about starting EM Express to manage a PDB.
- See Managing PDBs with EM Express for information on the features provided in EM Express for managing PDBs. Many of those features are available on the Containers page for a CDB.

EM Express provides support for in-memory database features introduced in Oracle Database 12*c* Release 1 (12.1.0.2):

- The Allocation Breakdown and Allocation History charts on the Memory Management page show In-Memory Area information.
- The amount of CPU consumed is broken out into regular CPU usage (CPU usage not related to in-memory operations) and in-memory CPU usage (CPU usage for in-memory operations), when applicable. This CPU breakdown is shown in several places:
  - On the Monitored SQL Execution Details page, wherever CPU activity is shown.
  - On the SQL Details page, in the Activity % column.

- On the Activity tab in the Performance Hub, when Resource Consumption by Wait Class is selected.

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

The following are changes in *Oracle Database 2 Day DBA* for Oracle Database 12*c* Release 1 (12.1.0.1).

New Features

- Oracle Enterprise Manager Database Express (EM Express) is the next-generation management tool for administration and management of individual databases that comes with Oracle Database 12*c* Release 1 (12.1).

  In previous Oracle Database releases, Oracle Enterprise Manager Database Control (DB Control) was the primary database management tool. DB Control is not included in this release.

  See "Introduction to Oracle Enterprise Manager Database Express" for more information about EM Express.

- SQL Developer is used to manage schema objects.

  Previous versions of this book described how to manage schema objects using DB Control. This version describes how to manage schema objects using SQL Developer.

  See Managing Schema Objects for more information about using SQL Developer.

- Oracle Recovery Manager (RMAN) is used to perform database backup and recovery operations.

  Previous versions of this book described how to perform database backup and recovery operations using DB Control. This version describes how to perform database backup and recovery operations using RMAN.

  See Performing Backup and Recovery for more information about using RMAN to perform database backup and recovery.

1

Introduction

As a database administrator (DBA), you are responsible for the overall operation of Oracle Database. This introductory chapter is intended to help orient you to many common DBA tasks, to the tools available to you, and to this guide. The chapter contains the following sections:

About This Guide

Oracle Database 2 Day DBA is a database administration quick start guide that teaches you how to perform day-to-day database administrative tasks. The goal of this guide is to help you understand the concepts behind Oracle Database, and to help you learn how to perform all common administrative tasks needed to keep the database operational. These tasks include configuring the database, managing memory and storage, managing users, managing database objects such as tables, performing basic troubleshooting, creating backups for your database, performance monitoring activities, and more.

The primary administrative interface used in this guide is Oracle Enterprise Manager Database Express (EM Express), featuring all the self-management capabilities introduced in Oracle Database.

This section includes these topics:

What This Guide Is Not

Oracle Database 2 Day DBA is task-oriented. The objective is to describe why and when administrative tasks must be performed. Where appropriate, it describes the concepts necessary to understand and complete a task, assuming the reader has no prior knowledge of the database. This guide is not an exhaustive discussion of all Oracle Database concepts. For this type of information, see *Oracle Database Concepts*.

Additionally, for a complete discussion of administrative tasks, see *Oracle Database Administrator's Guide.*

How to Use This Guide with Related Material

This guide is part of a comprehensive set of learning material for administering Oracle Database, which includes a 2 Day DBA Oracle By Example (OBE) series, available on the Web.

Every chapter in *Oracle Database 2 Day DBA* has an associated Oracle By Example lesson. The OBE steps through tasks in the chapter and includes annotated screenshots. In some cases, the OBE provides additional information to help you complete the task.

At the end of each chapter, you can find the link to its associated OBE lesson. The home page for the 2 Day DBA Oracle By Example series is:

```
https://apex.oracle.com/pls/apex/f?
p=44785:24:0::::P24_CONTENT_ID,P24_PREV_PAGE:6282,1
```

About Oracle Database

Oracle Database is a relational database with object and Extensible Markup Language (XML) capabilities. In a relational database, all data is stored in tables that are composed of rows and columns. Oracle Database enables you to store data, update it, and efficiently retrieve it, with a high degree of performance, reliability, and scalability.

Oracle Database is composed of the following elements:

- The Oracle software that you install on your host computer
- The **database**, which is a collection of physical files on one or more disks

  The database contains user data, metadata, and control structures. **Metadata**, or data about the data, is the collection of information on disk that permits Oracle software to manage user data. An example of metadata is the data dictionary. Control structures (such as the control file and online redo log files) ensure the integrity, availability, and recoverability of user data.
- The **Oracle instance**, which is composed of the following:
  - The **background processes**, which are the operating system processes or threads that perform the work of accessing, storing, monitoring, and recovering user data, metadata, and control files associated with the database
  - The shared memory areas used by the background processes
- **Server processes** that perform work on behalf of connected users and applications, and the memory and temporary storage used by these processes

  Server processes parse and execute SQL statements, and retrieve and return results to the user or application.
- Oracle Net, which is a software layer that enables client applications and Oracle Database to communicate over a network, and the Oracle Net **listener**, which is a process that listens for connection requests from the network.

See Also:

- Configuring the Network Environment
- Managing the Oracle Instance
- Managing Database Storage Structures
- Administering User Accounts and Security
- *Oracle Database Concepts* for more information about background processes
- *Oracle Database Reference* for more information about background processes

Common Oracle DBA Tasks

As an Oracle database administrator (DBA), you can expect to be involved in the following tasks:

- Installing Oracle software
- Creating Oracle databases
- Performing upgrades of the database and software to new release levels
- Starting and shutting down the database instance
- Managing the storage structures of the database
- Managing users and security
- Managing database objects, such as tables, indexes, and views
- Backing up the database and performing recovery operations when necessary
- Monitoring the state of the database and taking preventive or corrective action as required
- Monitoring and tuning database performance
- Diagnosing and reporting critical errors to Oracle Support Services

In a small to medium-sized database environment, you might be the sole person performing these tasks. In large, enterprise environments, the job is often divided among several DBAs, each of whom has a specialty, such as database security or database tuning.

See Also:

Oracle Database Concepts for more information about the duties of database administrators

Tools for Administering the Database

The goal of this guide is to enable you to quickly and efficiently create an Oracle database, and to provide guidance in basic database administration.

The following are some products, tools, and utilities you can use to achieve your goals as a database administrator:

- **Oracle Universal Installer**

  Oracle Universal Installer (OUI) is a utility that installs your Oracle software and options. It can automatically start Oracle Database Configuration Assistant to install a database.

- **Oracle Database Configuration Assistant**

  Oracle Database Configuration Assistant (DBCA) is a utility that creates a database from templates that are supplied by Oracle, or you can create your own. It enables you to copy a preconfigured seed database, thus saving the time and effort of generating and customizing a new database.

- **Database Upgrade Assistant**

  The Database Upgrade Assistant (DBUA) is a tool that guides you through the upgrade of your existing database to a new Oracle Database release.

- **Net Configuration Assistant**

  Net Configuration Assistant is a utility that enables you to configure listeners and naming methods, which are critical components of the Oracle Database network.

- **Oracle Enterprise Manager Database Express**

  The primary product for managing your database is Oracle Enterprise Manager Database Express (EM Express), a Web-based interface. After you have installed the Oracle Database software, created or upgraded a database, and configured the network, you can use EM Express to manage your database. EM Express also provides an interface for performance advisors.

  Oracle also offers separately licensed Oracle Enterprise Manager management packs, management plug-ins, and other products you can purchase to enhance the capabilities of Oracle Enterprise Manager in specific environments.

- **SQL Developer**

  SQL Developer provides another GUI for accessing your Oracle database. SQL Developer supports development in both the SQL and PL/SQL languages. It is available in the default installation of Oracle Database.

  With SQL Developer, you can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. You can also run any number of provided reports, and also create and save your own.

See Also:

Oracle Database Licensing Information

2

Installing Oracle Database and Creating a Database

This chapter describes how to install Oracle Database software and create a single instance Oracle Database.

If you are using an earlier release of Oracle Database and want to install a later release of the Oracle Database software, then you can *upgrade* your existing Oracle Database and use it with the new release of the database software. See "Upgrading a Database".

This chapter contains the following sections:

Note:

This chapter provides an overview of how to install Oracle Database software and create a single instance Oracle Database. This chapter is not a complete installation guide for Oracle Database.

For more detailed information about installing Oracle Database software, see *Oracle Database Installation Guide* for your platform.

Overview of Installing Oracle Database Software and Creating a Database

To install your Oracle Database software, use Oracle Universal Installer (OUI). OUI is a graphical user interface utility that enables you to install new Oracle Database software. Online Help is available to guide you through the installation process.

During the installation process, you are given the opportunity to create a database. If you choose to do so, then OUI automatically starts Oracle Database Configuration Assistant (DBCA) to guide you through the process of creating and configuring a database.

Before you start the installation process, see the following sections for information about prerequisites and installation choices:

If you do not create a database during installation, then you must run DBCA at some point after installation to create a database.

Note:

After you create a database, either during installation or as a standalone operation, you do not have to create another. Rather than requiring that you create multiple databases to accommodate different applications, you can separate data into different *schemas* within a single Oracle Database. See "About User Accounts" for more information about schemas.

Starting with Oracle Database 12*c*, it is also possible to create a multitenant container database (CDB) that can support zero, one, or many user-created pluggable databases (PDBs). All Oracle databases created before Oracle Database 12*c* are non-CDBs. This manual describes the OUI and DBCA options for creating CDBs and PDBs, and subsequent chapters provide information on managing CDBs and PDBs. See *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for more information about CDBs and PDBs.

Checking Oracle Database Installation Prerequisites

Before installing the software, Oracle Universal Installer (OUI) performs several automated checks to ensure that your computer fulfills the basic hardware and software requirements for an Oracle Database installation. If your computer does not meet a requirement, then an error message is displayed. The requirements may vary depending upon the type of computer and operating system you are using, but some prerequisites include:

- There is a minimum of 1 GB of physical memory.
- Sufficient paging space is available.
- The appropriate service packs or patches for your operating system are installed.
- An appropriate file system format is being used.

See Also:

Oracle Database Installation Guide for your platform for more information about preinstallation requirements and tasks

Deciding on Oracle Database Installation Choices

Oracle Universal Installer (OUI) guides you through an interview phase where you specify your choices for installation and database creation. The exact sequence of steps depends on your operating system. As you progress through the installation, you are presented with choices on how to configure the database.

- Install Option for Oracle Database
- Installation Method for Oracle Database
- Installation Type for Oracle Database

- Software Installation Directories for Oracle Database
- Database File Location for Oracle Database
- Database Identifiers for Oracle Database
- About Advanced Installation for Oracle Database

Install Option for Oracle Database

You can choose to create and configure a database, or to only install the database software.

You can create a preconfigured database or a custom-configured database during installation. If you choose not to create a database during installation, then you must run Database Configuration Assistant (DBCA) after installation to create a database.

Note:

If you choose to create and configure a database, then Oracle Universal Installer (OUI) will start DBCA at the end of the installation to configure the database.

If you choose to only install the database software using OUI, then you must manually run DBCA after the installation to create and configure the database. With this approach, more options are available for controlling database configuration.

Preconfigured databases are based on templates that Oracle provides or that you create. Each Oracle-provided template is optimized for a particular workload type. See Table 2 for information about the types of preconfigured databases.

If you choose to use the Desktop Class installation method, then the general purpose database template is used. To create a custom database in which you configure your own database structure, see "About Advanced Installation for Oracle Database."

Note:

If you must create a new database, then Oracle recommends that you install a preconfigured database, which is faster and easier. You can customize the database after it has been created.

Installation Method for Oracle Database

The installation methods are divided into Desktop Class and Server Class:

- **Desktop Class**—This installation class is most appropriate for laptop or desktop computers. It includes a starter database and requires minimal configuration.
- **Server Class**—This installation class is for servers, such as you would find in a data center, or used to support enterprise-level applications. Choose this installation class if you need access to advanced configuration options.

During a Desktop Class installation, you make only basic choices. For a Server Class installation, you choose either typical installation (where you make only basic choices) or advanced installation.

During a Desktop Class or a typical installation, Oracle Database automatically installs the sample schemas.

Installation Type for Oracle Database

When you install Oracle Database during basic and advanced installations, you need answers for the questions listed in this section. Oracle Universal Installer (OUI) provides default values for every choice.

- What type of database edition installation do you want to perform?

  Your choices are:

  - **Enterprise Edition**—This installation type is the full-featured Oracle Database product that provides data management for enterprise-level applications. It is intended for mission-critical, high-security online transaction processing (OLTP) and data warehousing environments.
  - **Standard Edition**—This installation type is suitable for workgroup or department-level applications, and for small to medium-sized enterprises. It provides core relational database management services and options and includes an integrated set of management tools, replication, Web features, and facilities for building business-critical applications.
  - **Standard Edition One**—This installation type is suitable for workgroup, department, or web applications. It provides core relational database management services for single-server environments or highly distributed branch environments. Oracle Standard Edition One includes all the facilities necessary to build business-critical applications.
  - **Personal Edition** (Microsoft Windows operating systems only)—This installation type installs the same software as the Enterprise Edition, but supports only a single-user, development and deployment environment.

- What are your database configuration options?

Software Installation Directories for Oracle Database

You must specify the directory in which the Oracle Database software is installed, or the location where the product binary files are copied from the installation media. You must choose a location that has enough disk space to contain the software and is accessible by the operating system user performing the installation.

You also specify the location of the Oracle base directory, which is used by all Oracle software products installed on the server. The first time you install Oracle software on a server, you are prompted to specify the location of the inventory directory, called `oraInventory`. This directory provides a centralized inventory of all Oracle software products installed on the server. You should use the same value for the Oracle inventory directory each time you perform an Oracle software installation on the server.

Database File Location for Oracle Database

A database includes several files that store the user data, database metadata, and information required to recover from failures. As an administrator, you decide what kind of storage subsystem to use for these files. You can select from the following options:

- **File System**—This default option creates database files that are managed by the file system of your operating system. You can specify the directory path where database files are to be stored. Oracle Database can create and manage the actual files.

  If you are not certain about which option to use, then select File System (the default).

- **Automatic Storage Management**—This option enables you to place your data files in Oracle Automatic Storage Management (Oracle ASM) disk groups. If you choose Oracle ASM, then Oracle Database automatically manages database file placement and naming. For environments with a large number of disks, this option simplifies database administration and maximizes performance. Oracle ASM performs software striping and mirroring at the file level for maximum storage flexibility, performance, and availability.

  Oracle ASM uses an Oracle ASM instance, which is distinct from the database instance, to configure and manage disk groups. A single Oracle ASM instance can provide storage for multiple databases on the same server.

  For more information, see *Oracle Automatic Storage Management Administrator's Guide*.

  Note:

  In past releases, Oracle ASM was installed as part of the Oracle Database installation. With Oracle Database 11*g* Release 2 (11.2), Oracle ASM is part of an Oracle Grid Infrastructure installation.

  To use Oracle ASM for storing database files, you must have installed Oracle ASM and created one or more disk groups before performing the Oracle Database installation.

Database Identifiers for Oracle Database

These options include your global database name and system identifier (SID). The **SID** is a unique identifier that is used to distinguish this instance from other Oracle Database instances that you may create later and run concurrently on your system.

The global database name is the full name of the database that uniquely distinguishes it from any other database. The global database name is in the form `database_name.database_domain`, for example `sales.example.com`. The database name portion `sales` is a simple name you call your database. The database domain portion `example.com` specifies the database domain in which the database is located. Together, the database name and domain form the global database name.

About Advanced Installation for Oracle Database

During advanced installations using the Server Class method you are prompted to make the additional choices listed in this section, and the choices for a typical installation. The installation process provides default values for every choice.

This guide describes, but does not document, these additional advanced installation choices. For more information, see *Oracle Database Installation Guide* for your platform.

- Product Languages

You choose which language the software should use after it is installed. You can select multiple languages. The default value is English. If you choose a value other than English, it does not change the language used by the installation.

- Database Configuration Type

  You select a template to use when configuring the database. You can choose either General Purpose/Transaction Processing or Data Warehousing.

- Database Configuration Options

  You can choose how to configure the database created by the installer. You can select the memory size and management options, the character sets used to store data, the security options for database access, and whether the sample schemas should be installed.

  To complete the exercises in this guide and related course material, you must install the sample schemas. This data is also used in most examples throughout Oracle Database documentation. Oracle recommends that you install the sample schemas.

  This choice is a configuration option only during advanced installation. Sample schemas are installed by default during typical or Desktop class installations.

- Recovery Options

  You specify whether automated backups should be configured for the database. If you choose this option, you must specify whether the recovery area should be stored on the local file system or in an Oracle ASM disk group. You must also specify the operating system credentials the backup job uses when performing backups.

  Note:

  To use Oracle ASM for recovery area storage, you must have installed Oracle ASM as part of an Oracle Grid Infrastructure installation and created one or more disk groups before performing the Oracle Database installation.

- Schema Passwords

  When you create a database, certain administrative user accounts are created automatically. You are prompted to enter the passwords for administrative accounts such as the `SYS` and `SYSTEM` accounts, which enable you to manage and administer the database. You can use the same password for each account, or specify passwords for each account individually. If you do not enter a secure password, you will receive a warning message during installation.

- Operating System Groups

  Administrative access to the database is granted by membership in certain operating system groups. You can choose the operating system group to be used for `SYSDBA` access (typically `dba`) and `SYSOPER` access (typically `oper`).

  The `SYSDBA` group identifies operating system user accounts that have database administrative privileges and can log in with `SYSDBA` access. The `SYSOPER` group is an optional group for users that should have limited database administrative privileges. See "SYSDBA and SYSOPER System Privileges" for more information about these groups and privileges.

Installing Oracle Database Software

This section briefly describes the steps for a desktop-class installation. Most steps are common to all platforms and involve running Oracle Universal Installer (OUI). Platform-specific steps are noted. For further assistance, consult the online Help or the *Oracle Database Installation Guide* for your platform.

Note:

The following steps describe the OUI workflow for a host computer that has no previous Oracle software installed. If your host computer has Oracle software installed, then you may see a different workflow.

To perform a basic installation:

1. Log on to your computer as a member of the administrative group that is authorized to install Oracle Database software and to create and run the database.

   Refer to your operating system-specific documentation or contact your system administrator to determine whether you have the necessary privileges and permissions to install new software.

2. Do one of the following:

   - If you are installing from distribution media, then insert the distribution media for the database into your computer.

     The Autorun feature opens the Select a Product to Install window automatically.

   - If you downloaded the installation software from the Oracle Web site, then follow the instructions on the site to run the Oracle Universal Installer. Or, see the *Oracle Database Installation Guide* for your platform.

3. The first window that appears is the Configure Security Updates window. To receive notifications about security issues via e-mail, enter your e-mail address in the Email text field. To receive security updates from My Oracle Support, enter the e-mail address registered with My Oracle Support, select the **I wish to receive security updates...** option, and enter your My Oracle Support password.

   Click **Next** to continue.

   The Select Installation Option window appears.

4. Choose the **Create and configure a database** option. Or, you also have the option of choosing to only install the database software, but then you must create a database in an additional step after the software is installed. If you are currently using a previous version of Oracle Database, choose **Upgrade an existing database**. After you have chosen an option, click **Next**.

   The System Class window appears.

5. Choose **Desktop Class**.

   You can choose the Server Class option to customize your installation. For example, you use this method to configure Oracle Automatic Storage Management

for your database, install the Sample Schemas, or configure automated backups. Selecting this option guides you through a series of installation steps that are not documented in this guide. For more information about the advanced choices, see "About Advanced Installation for Oracle Database". Also see *Oracle Database Installation Guide* for your platform.

Click **Next**.

The Typical Install Configuration window appears.

6. Provide the following configuration details for the database:

   - Oracle Base— The Oracle base directory helps to facilitate the organization of multiple Oracle software installations. See *Oracle Database Installation Guide* for your platform for more information about `ORACLE_BASE`.

     If you did not set the `ORACLE_BASE` environment variable before starting OUI, then the Oracle base directory is created in an `app/username/` directory on the first existing and writable directory from `/u01` through `/u09` for UNIX and Linux systems, or on the disk drive with the most available space for Windows systems. If `/u01` through `/u09` does not exist on the UNIX or Linux system, then the default location is *`user_home_directory`*`/app/`*`username`*.

     You can click **Browse** to find the directory you want to act as the Oracle base directory.

   - Software Location—The software location is the Oracle home for your database. You must specify a new Oracle home directory for each new installation of Oracle Database software. By default, the Oracle home directory is a subdirectory of the Oracle base directory.

     You can click **Browse** to find the directory where you want to install the Oracle Database software.

   - Database File Location—The database file location is the location where Oracle Database files are stored. By default, this location is *`Oracle_base`*`/oradata`. You can click **Browse** to select a different location.

   - Database Edition—Select either **Enterprise Edition**, **Standard Edition**, **Standard Edition One**, or **Personal Edition** (Microsoft Windows platforms only). See "Installation Type for Oracle Database".

   - Character Set—Choose the character set to use to store the data within the database. You can choose between the **Default**, which is based on the operating system language settings, or **Unicode**.

   - OSDBA Group (Linux and UNIX platforms only)—Specify the operating system DBA group. Host computer users in this group have administrative privileges on the database. This group is typically named **`dba`**. Refer to *Oracle Database Installation Guide for Linux* or for your UNIX platform for more details.

   - Global Database Name—Enter the fully qualified global database name. See "Installation Type for Oracle Database" for more on global database name.

   - Administrative Password—Specify the initial password for administrator accounts such as the `SYS` and `SYSTEM` accounts. If the password you choose is not a secure password, a warning message will be displayed.

- **Create as Container database**: Enable this option to create the database as a multitenant container database (CDB) that can support zero, one, or many user-created pluggable databases (PDBs).

  If you want Database Configuration Assistant (DBCA) to create a PDB when it creates the CDB, specify the PDB name in the **Pluggable database name** field.

After you enter the required information, click **Next**.

Note:

On Microsoft Windows operating systems only, the Specify Oracle Home User window appears. This window enables you to use a non-administrator, low privileged Windows User Account as the Oracle Home User. This option is recommended for database installation to ensure that Oracle services run with limited privileges. For single instance databases, you can also choose to allow the Oracle Installer to create a new Windows User Account (local user only) which will then be used as the Oracle Home User.

If you decline this option, all the services will be installed and will run as the System user.

See *Oracle Database Platform Guide for Microsoft Windows* for more information about this feature.

If Oracle software has not previously been installed on this server, then the Create Inventory window appears. If this is not the first installation attempt on this server, then the Perform Prerequisite Checks window appears.

7. *For first time installations on Linux and UNIX operating systems only*, specify a directory for installation files and the name of an operating system group that has write permissions for that directory.

   If this is the first time you are installing any Oracle software on this computer, then the Create Inventory Directory window appears. You must specify a local directory for the inventory, which OUI uses to keep track of all Oracle software installed on the computer. This information is used while applying patches or upgrading an existing installation, and while deinstalling Oracle software. Note that this directory is different from the Oracle home directory. The recommended value for the inventory directory is *`Oracle_base`*`/../oraInventory`, or one level above the Oracle base directory, in the `oraInventory` subdirectory. If your Oracle base directory is `/u01/app/oracle`, then the Oracle inventory directory defaults to `/u01/app/oraInventory`.

   In this window you can also specify the operating system group that has write permissions on the inventory directory. This prevents other users from writing over the Oracle product installation files. Typically the `oinstall` group is selected.

   After you enter a directory path and specify an operating system group, click **Next** to continue.

   The Perform Prerequisite Checks window appears.

8. If any checks failed, then take corrective actions.

   OUI performs several environment checks and indicates whether the check was a success, or resulted in a warning or failure. Details of the checks are provided in

the displayed window. The installation can proceed only when all checks have a status of either Succeeded or Warning. If any of the environment checks failed, then they must be resolved manually. See "Checking Oracle Database Installation Prerequisites" for more information.

If all the prerequisite checks passed, or after you click **Next**, the Summary window appears,

9. Review the installation summary, then click **Finish** to start the installation.

   The Install window appears, showing the installation progress. After the installation phase, the Configuration Assistants window appears. This window lists the configuration assistants that are started automatically. If you chose to create a starter database, then DBCA starts automatically in a separate window.

   After database creation, a window is displayed that summarizes the database creation.

10. (Optional) Click **Password Management** to unlock user accounts to make the accounts accessible to users.

    The `SYS` and `SYSTEM` accounts are unlocked by default.

11. Click **OK** to continue the installation.

12. *For Linux and UNIX operating systems only*, run the specified scripts, then click **OK**.

    In the Execute Configuration Scripts window, you are prompted to open a new terminal window, and to run scripts as the `root` user. After you run the scripts, return to this window and click **OK**.

13. Make note of the information in the Finish window, then click **Close** to exit OUI.

    Your installation and database creation is now complete.

    You use Oracle Enterprise Manager Database Express (EM Express) to perform common database administration tasks.

    Use the URL for EM Express that is provided in the Finish window to start EM Express, specifying your database hostname instead of 'localhost.' When EM Express prompts you for your username and password, log in as a user with DBA privilege (such as SYSTEM).

Note:

By default, DBCA picks a free port from the 5500 to 5599 range to use as the EM Express port.

If you want a particular port to be used as the EM Express port, specify that port using the `DBEXPRESS_HTTPS_PORT` operating system environment variable prior to starting OUI or DBCA.

For more information on setting environment variables, see "Configuring the Operating System Environment Variables ."

See Also:

- *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for more information about CDBs and PDBs
- "SYS and SYSTEM Users" for information about the recommended alternative to using the `SYSTEM` account for day-to-day administrative tasks
- Getting Started with Database Administration for more information about using EM Express.

Creating and Managing a Database with DBCA

Unless you specified that only the Oracle Database software should be installed, Oracle Universal Installer (OUI) automatically runs Database Configuration Assistant (DBCA) after software installation is complete. DBCA then creates a database using the information you provided. If you do not create a starter database and later want to create one, or to create additional databases, use DBCA.

Note:

With Oracle Database, you typically have a single database that hosts multiple applications. You do not need multiple databases to run different applications. Instead, you can separate the objects that support each different application into different *schemas* in the same database. However, there may be situations in which you want to create multiple Oracle databases on the same host computer. When you do this with DBCA, the new databases typically use the same Oracle home directory as the first database, but store database data files separately from those of the first database.

DBCA also enables you to modify a database configuration, delete a database, and more. You can perform the following DBCA tasks:

- Starting DBCA
- Creating a Database Using DBCA
- Changing the Configuration of a Database Using DBCA
- Deleting a Database Using DBCA
- Managing Templates with DBCA
- Using DBCA to Manage PDBs

Online Help is available by clicking **Help**. It provides information that guides you in selecting configuration options.

Starting DBCA

Follow the steps in this section to start Database Configuration Assistant (DBCA).

Note:

If you choose to create a starter database while installing the Oracle Database software, then Oracle Universal Installer (OUI) automatically starts DBCA.

To start DBCA:

1. Log on to your computer as a member of the administrative group that is authorized to install Oracle Database software and to create and run the database.
2. Do one of the following:
   - To start DBCA on a Microsoft Windows operating system, click **Start**, select **Programs** (or **All Programs**), then **Oracle - *HOME_NAME***, then **Configuration and Migration Tools**, and then **Database Configuration Assistant**.
   - To start DBCA on UNIX or Linux, or at the command-line prompt on the Windows operating system, enter the following command:

     ```
     dbca
     ```

     The `dbca` utility is typically located in the *`ORACLE_HOME`*`/bin` directory.

Creating a Database Using DBCA

Database Configuration Assistant (DBCA) enables you to create an Oracle database by following a step-by-step guided workflow.

To create a database using DBCA:

1. Start DBCA as described in "Starting DBCA".
2. In the Database Operation window, select **Create Database** and click **Next** to start the guided workflow for creating a database. If you then select **Advanced Mode** and click **Next**, the workflow requests your input in the following windows:

- DBCA Creation Mode Window
- DBCA Database Template Window
- DBCA Database Identification Window
- DBCA Management Options Window
- DBCA Database Credentials Window
- DBCA Network Configuration Window
- DBCA Storage Locations Window
- DBCA Database Options Window
- DBCA Initialization Parameters Window
- DBCA Creation Options Window
- DBCA Prerequisite Steps Window

- DBCA Summary Window
- DBCA Progress Window

Most of these windows provide default settings. Depending on the options you choose in DBCA, some of these windows may not be displayed.

DBCA Creation Mode Window

The Database Configuration Assistant (DBCA) Creation Mode window enables you to create a database with default configuration or to use Advanced Mode to create a database.

If you choose **Advanced Mode**, you can customize storage locations, initialization parameters, management options, database options, and different passwords for Administrator user accounts.

If you choose **Create a database with default configuration**, you make fewer choices in the options for your database, which allows you to create your database sooner.

When you select **Create a database with default configuration**, you can select the following options:

- **Global Database Name**: Enter the database name in the form *database_name.domain_name*.
- **Storage Type**: Choose either **File System** or **Automatic Storage Management**.

  When you choose **File System**, your database files are managed by the file system of your operating system.

  When you choose **Automatic Storage Management**, you place your data files in Oracle Automatic Storage Management (Oracle ASM) disk groups.
- **Database Files Location**: The choice you make for the **Storage Type** option determines what you specify for the **Database Files Location** option.

  When you choose **File System** in the **Storage Type** field, you specify the directory path where the database files are to be stored in the **Database Files Location** field. Oracle Database can create and manage the actual files.

  When you choose **Automatic Storage Management** in the **Storage Type** field, you specify the disk group to use in the **Database Files Location** field (the disk group must already exist). With Oracle ASM, Oracle Database automatically manages database file placement and naming.
- **Fast Recovery Area**: Specify a backup and recovery area.
- **Database Character Set**: Choose the character set to use for the database. See "Character Sets" for more information about character sets.
- **Administrative Password**: Enter the password to use for the database administrative passwords (such as the `SYS` and `SYSTEM` accounts).
- **User "Oracle Home User" Password** *(on Microsoft Windows operating systems only)*: If during the installation you specified a non-administrator, low privileged Windows User Account (as Oracle Home User) to run the database services under, you are prompted for the password of that user account.
- **Create as Container Database**: Enable this option to create the database as a multitenant container database (CDB) that can support zero, one, or many user-created pluggable databases (PDBs).

If you want DBCA to create a PDB when it creates the CDB, specify the PDB name in the **Pluggable Database Name** field.

See Also:

- *Oracle Database Platform Guide for Microsoft Windows* for more information about the Oracle Home User feature
- *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for more information about CDBs and PDBs

DBCA Database Template Window

The Database Configuration Assistant (DBCA) Database Template window enables you to select the type of database template to use to create the database. You can select:

- **General Purpose or Transaction Processing**
- **Custom Database**
- **Data Warehouse**

Oracle Enterprise Manager Database Express (EM Express) supports Oracle single instance databases, including non-CDBs, multitenant container databases (CDBs), and pluggable databases (PDBs).

You can use DBCA to create a database from templates supplied by Oracle or from templates that you create. The templates contain settings optimized for a particular type of workload.

Oracle ships templates for the following two workload types:

- General purpose or transaction processing
- Data warehouse

Select the template suited to the type of workload your database will support. If you are not sure which to choose, then select the default **General Purpose or Transaction Processing** template.

Note:

The General Purpose or Transaction template and the Data Warehouse template create a database with the `COMPATIBLE` initialization parameter set to `12.1.0.2.0`.

For more complex environments, you can select the **Custom Database** option. This option does not use templates and results in a more extensive interview, which means that it takes longer to create your database.

See Also:

- *Oracle Database 2 Day + Real Application Clusters Guide* for more information about Oracle Real Application Clusters databases
- *Oracle Database Reference* for more information about the `COMPATIBLE` initialization parameter
- "Managing Templates with DBCA" for more information about using database templates

DBCA Database Identification Window

In the **Global Database Name** field of the Database Configuration Assistant (DBCA) Database Identification window, enter the database name in the form *database_name.domain_name*.

In the **SID** field, enter the system identifier. The SID defaults to the database name and uniquely identifies the instance that runs the database.

If you enable the **Create as Container Database** option, the database is created as a multitenant container database (CDB) that can support zero, one, or many user-created pluggable databases (PDBs).

If you do not want DBCA to create a PDB when it creates the CDB, enable the **Create an Empty Container Database** option.

If you want DBCA to create one or more PDBs when it creates the CDB, enable the **Create a Container Database with one or more PDBs** option. Then enter the number of PDBs to create in the **Number of PDBs** field. In the **PDB Name** field, specify the name to use for the PDB or PDBS to be created. When you create multiple PDBs, the PDB name you specify is used as a prefix for the PDBs to be created. For example, if you ask for 3 PDBs to be created and specify SANDBOXPDB as the PDB name, then the names of the PDBs created will be SANDBOXPDB1, SANDBOXPDB2, and SANDBOXPDB3.

See Also:

Oracle Database Concepts and *Oracle Database Administrator's Guide* for more information about CDBs and PDBs

DBCA Management Options Window

Use the Database Configuration Assistant (DBCA) Management Options window to set up your database so it can be managed with Oracle Enterprise Manager. Oracle Enterprise Manager provides Web-based management tools for individual databases, and central management tools for managing your entire Oracle environment.

- To manage your database locally, select **Configure Enterprise Manager (EM) Database Express**.
- If the Oracle Management Agent is installed on your host computer, then you can choose central management by selecting **Register with Enterprise Manager (EM) Cloud Control for centralized management** and then specifying the host and port for the Management Service and the EM Admin username and password.

See Also:

Oracle Database Administrator's Guide for more information about Oracle Enterprise Manager Cloud Control

DBCA Database Credentials Window

In the Database Configuration Assistant (DBCA) Database Credentials window, specify the passwords for the administrative accounts such as `SYS`, `SYSTEM`, and `PDBADMIN`.

On Microsoft Windows operating systems only: If during the installation you specified a non-administrator, low privileged Windows User Account as Oracle Home User, you are prompted for the password of that user account.

DBCA Network Configuration Window

In the Database Configuration Assistant (DBCA) Network Configuration window, the listeners in the current Oracle home are displayed. If you want to create a new listener in the current Oracle home, you can do it in the Network Configuration window.

DBCA Storage Locations Window

In the Database Configuration Assistant (DBCA) Storage Locations window, specify the type of storage you would like your database to use.

If you choose **File System**, your database files are managed by the file system of your operating system.

If you choose **Automatic Storage Management (ASM)**, you place your data files in Oracle Automatic Storage Management (Oracle ASM) disk groups.

Then specify the locations for the Oracle database files. Select one of the following options:

- **Use Database File Locations from Template**—This option instructs DBCA to use the directory information as specified in the template. Later, you can make modifications to database file names and locations.
- **Use Common Location for All Database Files**—This option requires you to specify a new directory for the Oracle home. All the database files are created in this location. Later, you can make modifications to database file names and locations.

  If you specify the **Use Oracle Managed Files** option, Oracle Database will directly manage operating system files comprising an Oracle database. You specify the default location, called a database area, for all your files. Oracle Database thereafter automatically creates and deletes files in this location, as required. When you select this option, you delegate the complete management of database files to the database. You no longer have to specify the file names, their location, or their sizes.

When you create a new database, it is important to configure the database so you can recover your data if a system failure occurs. Online redo log files contain a record of changes that were made to data files. Online redo log files are stored in online redo log groups. You must have at least two online redo log groups for your database. After the online redo log files in a group have filled up, the log writer process (LGWR) switches the writing of redo records to a new online redo log group. Oracle Database can automatically save the inactive group of online redo log files to one or more offline destinations, known collectively as the **archived redo log** (also called the archive log).

The process of turning online redo log files into archived redo log files is called **archiving**.

Archiving can be performed only if the database is running in `ARCHIVELOG` mode. A group of online redo log files cannot be reused by the log writer (LGWR) process until the group is archived. If the database is running in `NOARCHIVELOG` mode, then when a group becomes inactive after the LGWR process switches to a new group, the inactive group is available for immediate reuse by the LGWR process.

The `NOARCHIVELOG` mode protects a database from instance failure but not from media failure. Only the most recent changes made to the database, which are stored in the online redo log files, are available for instance recovery. To restore a database operating in `NOARCHIVELOG` mode, you can use only entire database backups taken while the database is closed. Therefore, if you operate a database in `NOARCHIVELOG` mode, then back up the entire database at regular, frequent intervals.

The archiving of online redo log files has the following advantages:

- A database backup, with online and archived redo log files, guarantees that you can recover all committed transactions if the operating system or hardware fails.
- You can recover the database using a backup that was taken while the database was open and being used, if you have a copy of the archived log files that were written while the database was being backed up.
- You can perform online tablespace backups, and use these backups to restore a tablespace following media failure.
- You can keep a standby database current with its original database by continuously applying the original archived redo log files to the standby database.

Before you can archive the online redo log files, you must determine the destination to which you want to archive. Oracle recommends that the archive log be stored in a fast recovery area because it can simplify backup and recovery operations for your database. A **fast recovery area** is a location in which Oracle Database can store and manage files related to backup and recovery. It is distinct from the database area, which is a location for the current database files (data files, control files, and online redo log files).

When creating your database, you can select the following options:

- **Storage Type**—Specify the type of storage you would like your database to use for recovery-related files. For more information, see "About Advanced Installation for Oracle Database".
- **Specify Fast Recovery Area**—Select this option to specify a backup and recovery area and its directory location and size. You can use variables to identify standard locations.
- **Enable Archiving**—Select this option to enable the archiving of database online redo log files, which can be used to recover a database. Selecting this option is the same as running the database in `ARCHIVELOG` mode.

  Oracle recommends you select **Enable Archiving**. Selecting this option provides better protection for your database for software or hardware failure. If you do not select this option now, then you can enable ARCHIVELOG mode later. See "Configuring Your Database for Basic Backup and Recovery".

DBCA Database Options Window

The Database Configuration Assistant (DBCA) Database Options window includes the Sample Schemas tab and the Database Vault & Label Security tab.

On the Sample Schemas tab, you can specify for the sample schemas to be created in your database, and for custom SQL scripts to be run after database creation:

- **Sample Schemas—**Select **Sample Schemas** to include the Sample Schemas (EXAMPLE) tablespace in your database. Oracle guides and educational materials contain examples based upon the Sample Schemas. Oracle recommends that you include them in your database.
- **Select a script—**In this field, you can optionally specify one or more SQL scripts to run after your database is created. Scripts are useful for performing postinstallation tasks, such as loading custom schemas. Note that if you choose to run scripts after installation, then your scripts must include a connection string that identifies the database. Click **Help** for more information.

On the Database Vault & Label Security tab, you can choose options related to Oracle Database Vault and Oracle Label Security.

See Also:

- *Oracle Database Vault Administrator's Guide* for more information about Oracle Database Vault
- *Oracle Label Security Administrator's Guide* for more information about Oracle Label Security

DBCA Initialization Parameters Window

The links in the Database Configuration Assistant (DBCA) Initialization Parameters window provide access to additional windows that enable you to change the default initialization parameter settings. These parameters fall into the following categories:

- Memory
- Sizing
- Character Sets
- Connection Mode

You can also click the **All Initialization Parameters** button at the bottom of the window to display a list of all the database initialization parameters and their current settings.

Memory

Use the Memory tab of the Database Configuration Assistant (DBCA) Initialization Parameters window to set the initialization parameters that control how the database manages its memory. You can choose from the following methods for memory management:

- **Typical**—This method requires little configuration, and allocates memory as a percentage of total overall physical system memory. Select **Typical** and enter a

value in the **Memory Size (SGA and PGA)** field. The percentage of memory represented by the value entered is shown in the **Percentage** field. You can also use the slider to change the value. Click **Show Memory Distribution** to view the System Global Area (SGA) size and Program Global Area (PGA) size allocated. Click **Use Automatic Memory Management** to have the system automatically tunes many of the memory components of the SGA, and allocate memory to individual PGAs as needed. The system can also dynamically decrease or increase the total amount of memory allocated to the SGA or aggregate PGA, depending on processing demands. The total memory used for the database instance never exceeds the amount you specify. This automatic memory tuning for the instance is known as **automatic memory management**. To learn more about PGA and SGA, see "About Instance Memory Structures".

- **Custom**—This method requires more configuration than the Typical option, but gives you more control over how the database instance uses system memory. This option is meant for more experienced database administrators. You can directly specify memory sizes for the SGA and aggregate PGA and their substructures, such as the shared pool and buffer cache.

  Select one of the following options for customized memory management:

  - **Automatic Shared Memory Management** to allocate specific amounts of memory to the SGA and aggregate PGA. With this setting, automatic shared memory management is enabled for the SGA, and memory is allocated to the individual PGAs as needed.
  - **Manual Shared Memory Management** to enter specific values for each SGA component and the aggregate PGA. This disables automatic shared memory management and enables you to determine how the SGA memory is distributed among the SGA memory components.

  See Also:

  Managing the Oracle Instance for more information about memory management options

Sizing

In the Sizing tab of the Database Configuration Assistant (DBCA) Initialization Parameters window, you specify the smallest block size and the maximum number of operating system user processes that can simultaneously connect to the database.

- **Block Size**—Use this list to select the block size, or accept the default. Oracle Database data is stored in data blocks of the size specified. One data block corresponds to a specific number of bytes of physical space on disk. Selecting a block size other than the default 8 kilobytes (KB) value requires advanced knowledge and should be done only when absolutely required.

  While using predefined templates, this list is not enabled because the database will be created with the default block size of 8 KB. If you chose **Custom Database** in the Database Type window, you can change the block size here.

- **Processes**—In this field, specify the maximum number of processes that can simultaneously connect to the database. Enter a number or accept the default value of 300. The default value for this parameter is appropriate for many environments.

The value you select should allow for all background processes, user processes, and parallel execution processes.

Character Sets

Use the Character Sets tab of the Database Configuration Assistant (DBCA) Initialization Parameters window to define the character sets used by your database. **Character sets** are the encoding schemes used to display characters on your computer screen. The character set determines what languages can be represented in the database.

Oracle recommends using Unicode (AL32UTF8) as the database character set. AL32UTF8 is Oracle's name for the UTF-8 encoding of the Unicode standard. The Unicode standard is the universal character set that supports most of the currently spoken languages of the world. The use of the Unicode standard is indispensable for any multilingual technology, including database processing.

After a database is created and accumulates production data, changing the database character set is a time consuming and complex project. Therefore, it is very important to select the right character set at installation time. Even if the database does not currently store multilingual data but is expected to store multilingual data within a few years, the choice of AL32UTF8 for the database character set is usually the only good decision.

If you create a multitenant container database (CDB), consider that the character set you select determines which other databases you can later plug into the CDB. Only databases with a compatible database character set can be plugged into the CDB.

The default character set used by Oracle Universal Installer (OUI) and DBCA for the UNIX, Linux, and Microsoft Windows platforms is not AL32UTF8, but a Microsoft Windows character set known as an ANSI code page. The particular character set is selected based on the current language (locale) of the operating system session that started OUI or DBCA. If the language is American English or a Western European language, then the default character set is WE8MSWIN1252. Each Microsoft Windows ANSI Code Page can store data from only one language or a limited group of languages, such as only Western European, or only Eastern European, or only Japanese.

A Microsoft Windows character set is the default even for databases created on UNIX and Linux platforms because Microsoft Windows is the prevalent platform for client workstations. Oracle Client libraries automatically perform the necessary character set conversion between the database character set and the character sets used by non-Windows client applications.

You may also choose to use any other character set from the presented list of character sets. You can use this option to select a particular character set required by an application vendor, or choose a particular character set that is the common character set used by all clients connecting to this database.

Because AL32UTF8 is a multibyte character set, database operations on character data may be slightly slower when compared to single-byte database character sets, such as WE8MSWIN1252. Storage space requirements for text in most languages that use characters outside of the ASCII repertoire are higher in AL32UTF8 compared to legacy character sets supporting the language. Note that the increase in storage space concerns only character data and only data that is not in English. The universality and flexibility of Unicode usually outweighs these additional costs.

- **Database Character Set—**In this section, select one of the following options:

- **Use the Default—**Select this option to select only the language currently used by the operating system for all your database users and database applications.
- **Use Unicode (AL32UTF8)**—Select this option to support multiple languages for your database users and database applications.
- **Choose from the list of character sets—**Select this option if you want Oracle Database to use a character set other than the default character set used by the operating system.

Note:

AL32UTF8 is a variable-width multibyte character set. Applications connecting to a database that uses AL32UTF8 for character data processing must be correctly programmed to work with such character sets. Always verify the character set requirements of the applications that use the database. Contact the application vendor and ask for a Unicode-capable version, if your current application version does not support the Unicode standard.

- **National Character Set—**In this list, select a character set or accept the default. The national character set, also called `NCHAR` character set, is the character set used to store and process data of data types `NVARCHAR2`, `NCHAR`, and `NCLOB`. These data types allow storing of Unicode characters in a database that does not have a Unicode database character set. Unless installation requirements of any of your applications specify otherwise, accept the default value of AL16UTF16 as the national character set.

Note:

Although this character set is called "national," after the SQL standard (ISO/IEC 9075), it is not better suited to support globalized applications than the database character set. Because working with national character set data requires additional API calls in client applications, and because national character set data is not supported by some database components, such as Oracle Text, Oracle recommends that multilingual applications use `VARCHAR2`, `CHAR`, and `CLOB` data types and an Oracle database with the database character set AL32UTF8.

- **Default Language**—In this list, select a default database language or accept the default. The default language determines how the database supports locale-sensitive information such as day and month abbreviations, default sorting sequence for character data, and reading direction (left to right or right to left).
- **Default Territory—**In this list, select the name of the territory whose conventions are to be followed for day and week numbering or accept the default. The default territory also establishes the default date format, the default decimal character and group separator, and the default International Standardization Organization (ISO) and local currency symbols. For example, in the United Kingdom, the default date format is DD-MON-YYYY, where DD is the day of the month (1-31), MON is the abbreviated name of the month, and YYYY is the 4-digit year.

Connection Mode

Use the Connection Mode tab of the Database Configuration Assistant (DBCA) Initialization Parameters window to select the database connection mode. You can run the database in either of the following connection modes:

- **Dedicated Server Mode**—This mode allows a dedicated server process for each user process. Select this option when the number of total clients is expected to be small, for example, 50 or fewer. You might also choose this option when database clients typically make persistent, long-running requests to the database. By default, the database is configured for dedicated server processes.
- **Shared Server Mode—**This mode allows several client connections to share a database-allocated pool of resources. Use this mode in configurations in which client load is expected to cause a strain on memory and other system resources. If you choose shared server mode, then you must indicate the number of server processes you want to create when a database instance is started. For more information about setting this parameter, click **Help**.

DBCA Creation Options Window

In the Database Configuration Assistant (DBCA) Creation Options window, select any of the following options for creating the database:

- **Create Database**—Select this option to create your database now.
- **Save as a Database Template**—Select this option to save the database definition as a template to use at a later time.
- **Generate Database Creation Scripts**—Select this option to generate a SQL database creation script that you can run at a later time.

DBCA Prerequisite Steps Window

The Database Configuration Assistant (DBCA) Prerequisite Steps window displays the results of the validation checks that DBCA performs to ensure that your system is capable of creating the database with the configuration options you have selected.

If any of the validation checks fail, Oracle recommends that you click **Back** to return to the previous window where you can fix the problem. However, you can also click **Ignore All** if you choose not to fix the problems that caused the failed validation checks.

DBCA Summary Window

The Database Configuration Assistant (DBCA) Summary window displays a summary of the configuration options that you have chosen for the database. Review the summary information.

To change any of these options, click **Back** and return to the window where you can modify the option.

Click **Finish** to have DBCA begin the creation of the database with the specified configuration options.

DBCA Progress Window

The Database Configuration Assistant (DBCA) Progress window displays the progress of the database creation operation.

When DBCA finishes, it displays the Database Configuration Assistant window, which advises you that database creation is complete.

The Database Configuration Assistant window provides information about:

- The location of the DBCA log files
- The global database name, SID, and server parameter file name for the database
- The URL to use to access Enterprise Manager to manage the database
- Managing the database accounts that were created

Changing the Configuration of a Database Using DBCA

You can use Database Configuration Assistant (DBCA) to change the configuration of an existing database. For example, you can make configuration changes such as:

- Add database options that were not previously configured (for example, Oracle Label Security or Oracle OLAP)
- Change default security settings
- Change the server mode from dedicated to shared, or the reverse

To change the configuration of a database using DBCA:

1. Start DBCA as described in "Starting DBCA".
2. In the Database Operation window, select **Configure Database Options** and click **Next**.
3. Follow the instructions in the DBCA guided workflow.

Deleting a Database Using DBCA

You can use Database Configuration Assistant (DBCA) to delete a database. When DBCA deletes a database, it shuts down the database instance and then deletes all database files. On the Windows platform, it also deletes associated Windows services.

To delete a database using DBCA:

1. Start DBCA as described in "Starting DBCA".
2. In the Database Operation window, select **Delete Database** and click **Next**.
3. Select the database to delete and click **Next**.

Managing Templates with DBCA

Database Configuration Assistant (DBCA) templates are XML files that contain information required to create a database. Templates are used in DBCA to create new databases and duplicate existing databases. The information in templates includes database options, initialization parameters, and storage attributes (for data files, tablespaces, control files, and online redo log files).

Templates can be used just like scripts, but they are more powerful than scripts because you have the option of duplicating a database. Duplication saves time because

you copy the files of an existing database, referred to as a *seed database*, to the correct locations.

Templates are stored in the following directory:

ORACLE_HOME`/assistants/dbca/templates`

Advantages of Using DBCA Templates

Using Database Configuration Assistant (DBCA) templates has the following advantages:

- Time saving. If you use a template, then you do not have to define the database.
- Easy duplication. By creating a template containing your database settings, you can easily create a duplicate database without specifying parameters twice.
- Easy editing. You can quickly change database options from the template settings.
- Easy sharing. Templates can be copied from one computer to another.

Types of DBCA Templates

Database Configuration Assistant (DBCA) templates are divided into the following types:

- Seed templates
- Nonseed templates

The characteristics of each are shown in Table 1 .

Table 1 DBCA Template Types

Type	File Extension	Includes Data Files	Database Structure
Seed	`.dbc`	Yes	This type of template contains both the structure and the physical data files of an existing database, referred to as the *seed database*. Your new database starts as a copy of the seed database, and requires only the following changes: • Name of the database • Destination of the data files • Number of control files • Number of online redo log groups • Initialization parameters Other changes can be made after database creation using custom scripts that can be invoked by DBCA, command-line SQL statements, or Oracle Enterprise Manager Database Express (EM Express). The data files for the seed database are stored in compressed Recovery Manager (RMAN) backup format in a file with a .dfb extension. The seed database control file is stored in a file with .ctl extension. (This file is needed only when storing data files in Oracle Automatic Storage Management (Oracle ASM) disk groups or as Oracle Managed Files.) The .dbc file contains the location of the seed database data files and contains the source database name used to mount the control file.

Type	File Extension	Includes Data Files	Database Structure
Nonseed	`.dbt`	No	This type of template is used to create a new database. It contains the characteristics of the database to be created. Nonseed templates are more flexible than their seed counterparts because all data files and online redo log files are created to your specification, and names, sizes, and other attributes can be changed as required.

DBCA Templates Provided by Oracle

Oracle provides the Database Configuration Assistant (DBCA) templates shown in Table 2 .

Table 2 Oracle-Provided DBCA Templates and Their Corresponding Workloads

Template	Workload
Data warehouse	Users perform numerous, complex queries that process large volumes of data. Response time, accuracy, and availability are key issues. These queries (`SELECT` statements) range from a fetch of a few records to queries that sort thousands of records from many different tables.
General Purpose or Transaction processing	Many concurrent users perform numerous transactions that require rapid access to data. Availability, speed, concurrency, and recoverability are key issues. Transactions consist of reading (`SELECT` statements), writing (`INSERT` and `UPDATE` statements), and deleting (`DELETE` statements) data in database tables.
Custom database	This template allows you maximum flexibility in defining a database because you can change any of the settings for the database being created.

Creating Templates Using DBCA

Follow the instructions in this section to create your own Database Configuration Assistant (DBCA) templates.

To create templates:

1. Start DBCA as described in "Starting DBCA".
2. In the Database Operation window, select **Manage Templates** and click **Next**.
3. In the Template Management window, select **Create a database template** and one of the following options, and click **Next**.

   - **From an existing template**

     Using an existing template, you can create a new template based on the predefined template settings. You can add or change any template settings such as initialization parameters, storage parameters, or whether to use custom scripts.

   - **From an existing database (structure only)**

You can create a new template that contains structural information from an existing database, including database options, tablespaces, data files, and initialization parameters. User-defined schemas and their data *will not* be part of the created template. The source database can be either local or remote. Select this option when you want the new database to be structurally similar to the source database, but not contain the same data.

- **From an existing database (structure as well as data)**

  You can create a new template that has both the structural information and physical data files of an existing database. Databases created using such a template are identical to the source database. User-defined schemas and their data *will* be part of the created template. The source database must be local. Select this option when you want a template from which you can create an exact replica of the source database.

When creating templates from existing databases, you can translate file paths into Optimal Flexible Architecture (OFA) or maintain existing file paths. OFA is a set of file naming and placement guidelines for Oracle software and databases. Using OFA is recommended if the target computer on which you plan to create a database using the template has a different directory structure than computer on which the template was defined. Standard file paths can be used if the target computer has a directory structure that is similar to the directory structure on the source computer.

4. Follow the instructions in the DBCA guided workflow to create your template.

Deleting DBCA Templates

When you delete a Database Configuration Assistant (DBCA) template, it is no longer available to create a new database or a new template.

To delete a template:

1. Start DBCA as described in "Starting DBCA".
2. In the Database Operation window, select **Manage Templates** and click **Next**.
3. In the Template Management window, select **Delete a database template** and click **Next**.
4. Select the template to delete and click **Next**.

Using DBCA to Manage PDBs

When a multitenant container database (CDB) exists, you can use Database Configuration Assistant (DBCA) to perform the following pluggable database (PDB) operations in the CDB:

- Create a PDB

  This option creates a new PDB in a CDB.

- Unplug a PDB

  This option unplugs a PDB. The unplugged PDB can be plugged into the same CDB or another CDB.

- Delete a PDB

This option deletes a PDB.

- Configure a PDB

  This option enables you to specify an Oracle Enterprise Manager Database Express (EM Express) port for the PDB, so that you can manage the PDB using EM Express. It also allows you to configure other database options for the PDB.

Managing PDBs in a CDB using DBCA

You can use Database Configuration Assistant (DBCA) to create, unplug, delete, or configure a pluggable database (PDB) in an existing multitenant container database (CDB).

Note:

The PDB operations can be performed only in a CDB. DBCA issues an error message if you attempt to perform PDB operations in a database that is not a CDB.

To manage PDBs using DBCA:

1. Start DBCA as described in "Starting DBCA".
2. In the Manage Pluggable Databases window, select one of the PDB operations and click **Next**.
3. In the Database List window, select the CDB in which to perform the selected PDB operation and click **Next**.
4. Follow the instructions in the DBCA guided workflow for the selected PDB operation.

See Also:

Oracle Database Concepts and *Oracle Database Administrator's Guide* for more information about CDBs and PDBs

---

Manually Installing the Database Sample Schemas Post-Installation

You may decide sometime after the initial database installation that you would like to install the database sample schemas. You can create the sample schemas manually by running SQL scripts.

See Also:

Oracle Database Sample Schemas for more information about creating the sample schemas manually using SQL scripts

---

Installation: Oracle By Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter and includes annotated screenshots.

To view the Installing Oracle Database Software and Creating a Database OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6281,1
```

3

Getting Started with Database Administration

This chapter provides a brief roadmap for administering your database. It introduces you to Oracle Enterprise Manager Database Express (EM Express), the Web-based interface for managing an Oracle database.

This chapter contains the following sections:

Managing Your Database: An Overview

This section provides an overview of the tasks involved in managing an Oracle database instance. Each chapter in this guide describes a different task in detail.

To manage your Oracle database:

1. Start the database instance.

   After the installation, your instance is started and your database is open. In the future, there will be times, perhaps for doing database maintenance or because of a power or media failure, that you shut down your database instance and later restart it.

   See "Shutting Down and Starting Up the Oracle Instance".

2. Optionally, configure the network environment to enable clients to connect to your database.

   See Configuring the Network Environment .

3. Review your database storage structures: tablespaces and data files, online redo log files, and control files. Create or modify storage structures as needed.

See Managing Database Storage Structures.

4. Review memory allocation and adjust as needed.

   See "Managing Memory".

5. Review, unlock, and reset passwords for predefined database users as needed. Create new users, and assign privileges and roles to them.

   See Administering User Accounts and Security.

6. Create the necessary schema objects, including tables, views, and indexes. Populate the tables with data.

   See Managing Schema Objects.

7. Create or review the backup strategy for the database and back up the database.

   See Performing Backup and Recovery.

8. Enable archiving of online redo log files, if not already done.

   See "Configuring Recovery Settings".

9. Monitor database performance, diagnose performance problems, and tune the database as necessary.

   See Monitoring and Tuning the Database.

10. Keep Oracle Database software up-to-date with the latest releases.

    See Managing Oracle Database Software .

Configuring the Operating System Environment Variables

Before using certain tools that access the Oracle database, such as SQL*Plus, you must configure environment variables for your operating system. These environment variables are used by Oracle Database to determine the database instance to which the tool should connect.

To configure operating system environment variables for your database instance on Linux and UNIX systems:

1. Open an operating system command window.
2. Ensure that the environment variables `ORACLE_HOME` and `ORACLE_SID` are set properly. The commands to use to set these environment variables depend on the shell you use to interface with the operating system. For example:

   - (bash or ksh) `export ORACLE_SID=orcl`
   - (csh or tcsh) `setenv ORACLE_SID orcl`

   You can set these with the scripts `coraenv` (for the C shell) and `oraenv` (for other shells). These scripts are typically located in the `/usr/local/bin` directory.

3. Ensure that the `$ORACLE_HOME/bin` directory is in your `PATH` environment variable.

4. You can also edit the profile file for your default shell in the home directory of the software owner, for example `/home/oracle`, so that these environment variables are set every time you log in as that user.

To configure operating system environment variables for your database instance on Windows systems:

1. Open an operating system command window.
2. Use either `regedit` or the Oracle Administration Assistant for Windows to make sure the `ORACLE_HOME` and `ORACLE_SID` parameters are set to the correct values in the `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_HOME_NAME` registry subkey.
3. Ensure that the `%ORACLE_HOME%\bin` directory is in your `PATH` environment variable. At a command prompt, use a command similar to the following:

```
set PATH=%ORACLE_HOME%\bin;%PATH%
```

See Also:

- *Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems* for more information about setting environment variables
- *Oracle Database Platform Guide for Microsoft Windows* for details on modifying the registry entries

Introduction to Oracle Enterprise Manager Database Express

Oracle Enterprise Manager Database Express (EM Express) is a web-based database management tool that is built inside the Oracle Database. It supports key performance management and basic database administration functions. From an architectural perspective, EM Express has no mid-tier or middleware components, ensuring that its overhead on the database server is negligible.

Using EM Express, you can perform administrative tasks such as managing user security and managing database memory and storage. You can also view performance and information about your database.

Note:

Enterprise Manager Database Control is no longer available in Oracle Database 12*c*. You can use Enterprise Manager Cloud Control 12*c* or EM Express 12*c* to manage your Oracle Database 12*c* databases.

Enterprise Manager Cloud Control supports Oracle Database 12*c* targets, including multitenant container databases (CDBs), pluggable databases (PDBs), non-CDBs, Oracle Real Application Clusters (Oracle RAC) databases, and Oracle Automatic Storage Management (Oracle ASM) databases.

See *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for more information about CDBs and PDBs, *Oracle Real Application Clusters Administration and Deployment Guide* for more information about Oracle RAC databases, and *Oracle Automatic Storage Management Administrator's Guide* for more information about Oracle ASM databases.

These Enterprise Manager Database Express features described below can be used against non-CDBs, CDBs, PDBs, or Oracle RAC database instances.

Configuration:

- Initialization parameters (init.ora) management
- Memory management
- Database Feature Usage
- Database Properties

Storage:

- Tablespace management
- Undo management
- Redo management[1]
- Archive log management[1]
- Control files management[1]

Performance:

- Performance Hub, which includes these features:
  - Real-time performance monitoring and tuning
  - Historical performance and tuning
  - SQL monitoring (real-time and historical)
  - Database operations monitoring
  - ADDM, including Real-Time ADDM
  - Active Session History (ASH) Analytics
- Automatic and manual SQL Tuning Advisor

Database Home Page

The main page for database administration is the Database Home page. This is the page that loads when you log in to EM Express. See "Accessing the Database Home Page".

Navigation

Menus at the top of the Database Home page organize database management tasks into distinct categories. Choosing a menu option takes you to the EM Express page for that database management task. For example, to view the Users page, from the **Security** menu, select **Users**.

[1] In a CDB, this feature is available for the CDB only, not for individual PDBs.

See Also:

Oracle Database Administrator's Guide for more information about Oracle Enterprise Manager Cloud Control

Starting EM Express

You can use Oracle Enterprise Manager Database Express (EM Express) to manage a non-CDB, a multitenant container database (CDB), or a pluggable database (PDB). For each non-CDB, CDB, or PDB that you want to manage using EM Express on a given host, a unique HTTPS port must be configured. A different port must be configured for each container in a CDB that you want to manage using EM Express.

Note:

You can use EM Express to manage a CDB, and all the PDBs in the CDB except for the seed PDB.

Usually the HTTPS port for a non-CDB, CDB, or PDB is provided by DBCA when it configures your non-CDB, CDB, or PDB.

You must know the HTTPS port for a non-CDB, CDB, or PDB to manage the database using EM Express.

When you specify the EM Express URL in your web browser, enter your database hostname instead of 'localhost.'

In other words, enter the EM Express URL in this format to start EM Express:

`https://`*`database-hostname`*`:`*`portnumber`*`/em/`

For example:

```
https://mydbhost.example.com:5500/em/
```

When EM Express prompts you for your username and password, log in as a user with `DBA` privilege (such as `SYS` or `SYSTEM`).

The "SYS and SYSTEM Users" topic provides information about the recommended alternative to using the `SYSTEM` account for day-to-day administrative tasks.

Note:

The first time you enter the URL for EM Express in your web browser, your browser may display warning messages.

EM Express is a servlet built on top of Oracle XML DB. The Oracle XML DB default wallet has a self-signed certificate, and some existing browsers consider self-signed certificates as untrusted because they are not signed by a trusted CA (certificate authority). However, the self-signed certificate is still secure, as it ensures that the traffic is encrypted between the Oracle XML DB server and the client (browser).

Therefore, enter a security exception for the EM Express URL in your web browser.

See Also:

SYS and SYSTEM Users

Starting EM Express for a Non-CDB

To start Oracle Enterprise Manager Database Express (EM Express) for a non-CDB, use the EM Express URL provided by Database Configuration Assistant (DBCA) when DBCA configured your non-CDB, as described in "Installing Oracle Database Software." That URL includes the HTTPS port number for the non-CDB.

If you do not know the HTTPS port number for the non-CDB, issue the following SQL statement in your non-CDB, and it returns the port that is configured for EM Express:

```
select dbms_xdb_config.gethttpsport() from dual;
```

If a port number is not returned by this statement, then you must manually configure an HTTPS port for this non-CDB, as described in "Configuring the HTTPS Port for EM Express."

See Also:

Oracle Database PL/SQL Packages and Types Reference for more information on the `gethttpsport` procedure

---

Starting EM Express for a CDB

To start Oracle Enterprise Manager Database Express (EM Express) for a multitenant container database (CDB), use the EM Express URL provided by Database Configuration Assistant (DBCA) when DBCA configured your CDB, as described in "Installing Oracle Database Software." That URL includes the HTTPS port number for the CDB.

If you do not know the HTTPS port number for the CDB, go to the root and issue the SQL statement that returns the port that is configured for EM Express:

```
alter session set container=CDB$ROOT;
select dbms_xdb_config.gethttpsport() from dual;
```

If a port number is not returned by this statement, then you must manually configure an HTTPS port for this CDB, as described in "Configuring the HTTPS Port for EM Express."

When connected to the root, EM Express displays data and enables actions that apply to the entire CDB.

See Also:

- Managing PDBs with EM Express for more information about using EM Express to manage the pluggable databases (PDBs) in a CDB
- *Oracle Database Administrator's Guide* for information about switching to a container using the `ALTER SESSION` statement
- *Oracle Database PL/SQL Packages and Types Reference* for more information on the `gethttpsport` procedure

---

Starting EM Express for a PDB

To start Oracle Enterprise Manager Database Express (EM Express) for a pluggable database (PDB), use the EM Express URL provided by Database Configuration Assistant (DBCA) when it configured the PDB. That URL includes the HTTPS port number for the PDB.

If you do not know the HTTPS port number for the PDB, go to the PDB you want to manage (PDB1 in this example), and issue the SQL statement that returns the port that is configured for EM Express:

```
alter session set container=PDB1;
select dbms_xdb_config.gethttpsport() from dual;
```

If a port number is not returned by this statement, then you must manually configure an HTTPS port for this PDB, as described in "Configuring the HTTPS Port for EM Express."

When connected to a PDB, EM Express displays data and enables actions that apply to the PDB only.

See Also:

- Managing PDBs with EM Express for more information about using EM Express to manage the PDBs in a multitenant container database (CDB)
- *Oracle Database Administrator's Guide* for information about switching to a container using the ALTER SESSION statement
- *Oracle Database PL/SQL Packages and Types Reference* for more information on the `gethttpsport` procedure

Configuring the HTTPS Port for EM Express

The steps in this section need to be performed only if Database Configuration Assistant (DBCA) did not provide you with the Oracle Enterprise Manager Database Express (EM Express) URL when configuring your database or pluggable database (PDB), or if you need to change the EM Express port later on. Otherwise, you can start EM Express by following the instructions in "Starting EM Express."

Before you can access EM Express from a Web browser, the HTTPS port for EM Express must be configured. After the HTTPS port for EM Express is configured, you use it to access EM Express.

To manually configure the HTTPS port for EM Express:

1. Configure and start the Oracle Net Listener (the listener). You can use lsnrctl to start, stop, and view the status of the listener.
2. If the listener is running on a nonstandard port (for example, not 1521), then the `init.ora` file for the database you want to manage using EM Express must contain a `local_listener` entry so that the HTTPS port can register with the correct listener. The `local_listener` entry references a TNSNAMES entry that points to the correct listener. For example:

   ```
   local_listener=inst1
   ```

where `inst1` is a TNSNAMES entry defined in `tnsnames.ora` that points to the listener. For example:

```
inst1= (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=host_name)(PORT=1234))
(CONNECT_DATA=(SERVICE_NAME=service_name)(SERVER=DEDICATED)))
```

In this example, 1234 is the nonstandard port on which the listener has been configured to listen.

3. Enable the TCP dispatcher by adding the following entry to the `init.ora` file for the database you want to manage using EM Express:

```
dispatchers="(PROTOCOL=TCP)(SERVICE=<sid>XDB)"
```

For example, if the database SID is ORCL, then the entry would be:

```
dispatchers="(PROTOCOL=TCP)(SERVICE=ORCLXDB)"
```

4. Restart the database so that the changes made in the `init.ora` file take effect.

5. Use the PL/SQL procedure `DBMS_XDB_CONFIG.SETHTTPSPORT` to set the HTTPS port for EM Express for the database to a port that is not in use. This will update the HTTPS `port` in the `xdbconfig.xml` file in the Oracle XML DB Repository. You must connect as `SYS / AS SYSDBA` to run the procedure.

For example, to set the HTTPS port for EM Express for a non-CDB:

```
SQL> exec DBMS_XDB_CONFIG.SETHTTPSPORT(5500);
```

To set the HTTPS port for EM Express for a multitenant container database (CDB), go to the root in the CDB and then use the PL/SQL procedure `DBMS_XDB_CONFIG.SETHTTPSPORT` in the CDB to set the HTTPS port for EM Express for the CDB to a port that is not in use. This will update the HTTPS `port` in the `xdbconfig.xml` file in the Oracle XML DB Repository. You must connect as `SYS / AS SYSDBA` to run the procedure. For example:

```
SQL> alter session set container=CDB$ROOT;
SQL> exec DBMS_XDB_CONFIG.SETHTTPSPORT(5501);
```

To set the HTTPS port for EM Express for a PDB, ensure that the PDB is open in read/write mode, and then use the PL/SQL procedure `DBMS_XDB_CONFIG.SETHTTPSPORT` in the PDB to set the HTTPS port for EM Express for the PDB to a port that is not in use. This will update the HTTPS `port` in the `xdbconfig.xml` file in the Oracle XML DB Repository. You must connect as `SYS / AS SYSDBA` to run the procedure. For example:

```
SQL> alter session set container=PDB1;
SQL> exec DBMS_XDB_CONFIG.SETHTTPSPORT(5502);
```

Use the following command to confirm that the port has registered with the listener:

```
$ lsnrctl status | grep -i 5502
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=hostname.example.com)
(PORT=5502)
)(Security=(my_wallet_directory=/$ORACLE_BASE/admin/sid/xdb_wallet))
(Presentation=HTTP)(Session=RAW))
```

6. To access EM Express for a non-CDB, CDB, or PDB, enter a URL in the following format in a Web browser, specifying the EM Express port number for the non-CDB, CDB, or PDB you want to manage:

```
https://database-hostname:portnumber/em/
```

For example:

```
https://mydbhost.example.com:5500/em/
```

When prompted for your username and password, log in as a user with DBA privilege (such as SYS or SYSTEM).

See Also:

- *Oracle Database Net Services Administrator's Guide* for more information about configuring and using the listener
- *Oracle Database Net Services Reference* for more information about listener parameters
- *Oracle Database Net Services Administrator's Guide* for more information about the `local_listener` entry
- *Oracle XML DB Developer's Guide* for more information about accessing the Oracle XML DB Repository
- *Oracle XML DB Developer's Guide* for more information about administering Oracle XML DB
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `sethttpsport` procedure
- *Oracle Database Administrator's Guide* for information about using the `ALTER PLUGGABLE DATABASE` statement to modify the mode of a PDB
- *Oracle Database Administrator's Guide* for information about switching to a container using the `ALTER SESSION` statement
- Managing PDBs with EM Express for more information about using EM Express to manage the PDBs in a CDB
- "SYS and SYSTEM Users" for information about the recommended alternative to using the `SYSTEM` account for day-to-day administrative tasks

Accessing the Database Home Page

The Database Home page is the main database management page in Oracle Enterprise Manager Database Express (EM Express).

To access the Database Home page:

1. Ensure that the HTTPS port for EM Express is configured.

   See "Configuring the HTTPS Port for EM Express".

2. In your Web browser, enter the EM Express URL for the database or pluggable database (PDB) you want to manage:

   ```
   https://database-hostname:portnumber/em
   ```

For example, if you installed the database on a host computer named `mydbhost.example.com`, and port number `5500` is configured as the HTTPS port number for EM Express for the database, then enter the following URL:

```
https://mydbhost.example.com:5500/em
```

If the database instance is running, then the Login page appears when you access EM Express.

If the database instance is not running, start it. See *Oracle Database Administrator's Guide* for information about starting the instance.

If port number 5501 is configured as the HTTPS port number for EM Express for the PDB you want to manage, then enter the following URL:

```
https://mydbhost.example.com:5501/em
```

If the PDB is open in read/write mode, then the Login page appears when you access EM Express.

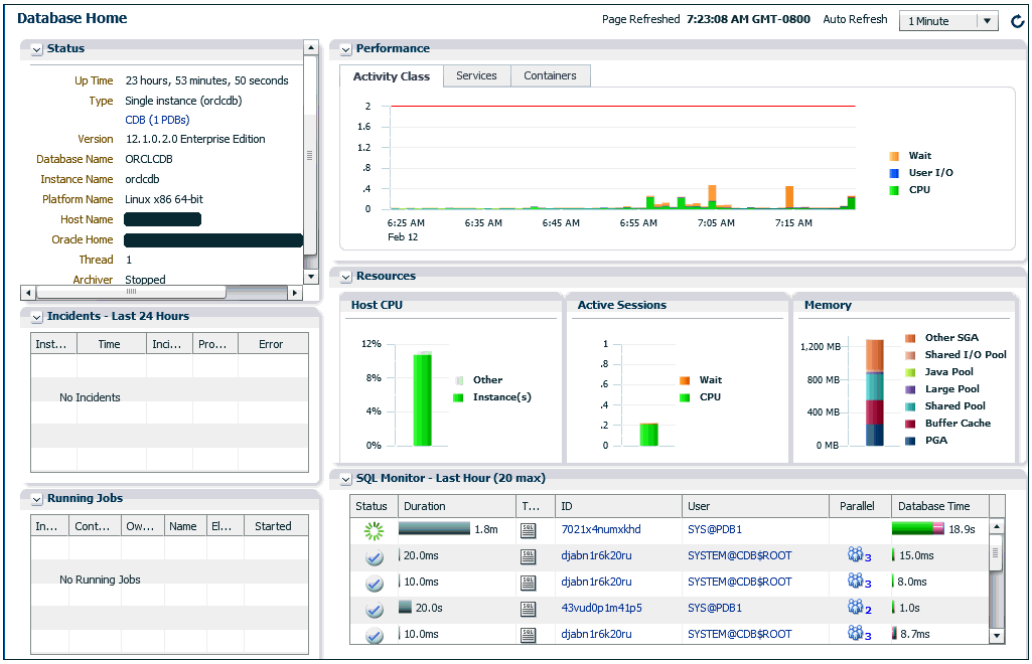
If the PDB is not open in read/write mode, open it in read/write mode. See *Oracle Database Administrator's Guide* for information about using the `ALTER PLUGGABLE DATABASE` statement to open a PDB in read/write mode.

3. Log in to the database or PDB with a user account that is authorized to access EM Express. The `EM_EXPRESS_BASIC` and `EM_EXPRESS_ALL` roles are created for EM Express, and a user who has been granted at least one of these roles can log in to EM Express.

   This user initially could be `SYS` or `SYSTEM`, with the password that you specified during database installation.

   Although the `SYSTEM` account can be used to perform day-to-day administrative tasks, Oracle strongly recommends creating a named user account for administering the Oracle database to enable monitoring of database activity. To back up, recover, or upgrade the database, you must log in as a user with the SYSDBA privilege.

   EM Express displays the Database Home page.



4. To return to the Database Home page after you have navigated to another EM Express page, you can:

   - Click the database icon at the top left of any EM Express page:

   - Click the **Back** button for your web browser until the Database Home page appears.

The various sections of the Database Home page provide information about the environment and status of the database. The Status section shows basic information about the database. When the database instance is a multitenant container database (CDB), the line after the Type field in the Status section is a link that identifies the instance as a CDB and lists the number of PDBs in the CDB. Click the **CDB (n PDBs)** link to view the Containers page for the CDB, which shows status, performance, and resource information for the CDB containers (PDBs). The Incidents - Last 24 Hours section lists critical error alerts in the database during the last 24 hours. The SQL Monitor section warns you of long-running SQL statements that may impact the performance of your database. Then, you can use the menu options to get more detail about the problem areas, and, in some cases, to obtain recommendations for resolving the problems. These topics are discussed in Monitoring and Tuning the Database.

See Also:

- "Configuring the HTTPS Port for EM Express"
- Managing PDBs with EM Express
- *Oracle Database 2 Day + Security Guide*

Granting Access to EM Express for Nonadministrative Users

As a database administrator, you can log in to Oracle Enterprise Manager Database Express (EM Express) with the SYS or SYSTEM user account to perform administrative and other tasks. Nonadministrative users may also want to log in to EM Express. For example, application developers may want to take advantage of the EM Express interface to create or modify tables, indexes, views, and so on. You must grant access to EM Express to these users before they can log in.

For nonadministrative users to have access to EM Express, they must be granted the EM_EXPRESS_BASIC or the EM_EXPRESS_ALL role.

The EM_EXPRESS_BASIC role enables users to connect to EM Express and to view the pages in read-only mode. The EM_EXPRESS_BASIC role includes the SELECT_CATALOG_ROLE role.

The EM_EXPRESS_ALL role enables users to connect to EM Express and use all the functionality provided by EM Express (read/write access to all EM Express features). The EM_EXPRESS_ALL role includes the EM_EXPRESS_BASIC role.

For an example of granting privileges and roles to a user account, see "Example: Granting Privileges and Roles to a User Account".

See Also:

- "SYS and SYSTEM Users" for information about the recommended alternative to using the SYSTEM account for day-to-day administrative tasks
- *Oracle Database 2 Day + Security Guide*

Administering the Database with SQL-Based Management Tools

In addition to using the graphical user interface (GUI) pages presented in Oracle Enterprise Manager Database Express (EM Express), you can use other Oracle tools such as SQL Developer and SQL*Plus to administer your database. These tools enable you to perform database management operations, and to query, insert, update, or delete data directly in the database.

The following sections provide details:

- About SQL
- About SQL*Plus
- Starting SQL*Plus and Connecting to the Database
- About SQL Developer

See Also:

- *Oracle Database 2 Day Developer's Guide*
- *Oracle Database SQL Language Reference*

About SQL

To perform many of its operations, Oracle Enterprise Manager Database Express (EM Express) submits structured query language (SQL) statements to the database. SQL (pronounced like *sequel*) is an industry-standard English-like computer programming language for querying and updating databases.

The following is an example of a SQL query that lists information about countries in a countries table, which is owned by user `hr`:

```
SELECT COUNTRY_ID, COUNTRY_NAME FROM HR.COUNTRIES;
```

SQL is a powerful language that can also be used to perform a variety of database administration tasks. The following SQL statement creates the database user `nick` and assigns him a password of your choosing, represented by *password*:

```
CREATE USER nick IDENTIFIED BY password;
```

When performing some administrative tasks in EM Express, you can click **Show SQL** to see the SQL statements that EM Express generates and submits.

About SQL*Plus

SQL*Plus is a command-line program that you use to submit SQL and PL/SQL statements to an Oracle database. You can submit statements interactively or as SQL*Plus scripts. SQL*Plus is installed with the database and is located in your *ORACLE_HOME*`/bin` directory.

You can start SQL*Plus from the command line, or on Microsoft Windows, from the Start menu.

When SQL*Plus loads, it issues the SQL prompt, which looks like this:

```
SQL>
```

At the SQL prompt, you can enter statements that perform administrative tasks such as shutting down the database or creating a new user, or you can query, insert, update, and delete data.

You can enter a single SQL statement on multiple lines. You must end each statement with a semicolon (;). For most statements, you can rerun a statement by entering a slash (/) on a line by itself.

See Also:

- *SQL*Plus User's Guide and Reference*
- *Oracle Database Concepts*

Starting SQL*Plus and Connecting to the Database

The section describes how to start SQL*Plus and connect to the database from both the command line and the Windows Start menu.

For a new installation, you connect to the database using either the `SYS` or `SYSTEM` database accounts. When you enter `SYS` or a slash (`/`) as the user name and provide the `AS SYSDBA` clause, your access is authenticated using operating system authentication. **Operating system authentication** uses your Windows, UNIX, or Linux

host user account to authenticate you to Oracle Database. You must have logged in to the host computer with a user account that is a member of a special host user group. On UNIX and Linux, this user group is typically `dba`. This type of authentication enables you to connect to an Oracle database that is not yet started, so that you can start it up. See *Oracle Database Administrator's Guide* for more information.

The following procedures show how to log in to the database as user SYS using the SYSDBA privilege.

To start SQL*Plus and connect to the database from the command line:

1. Open a command window.
2. Configure the operating system environment variables, as described in "Configuring the Operating System Environment Variables ."
3. Start SQL*Plus using a command in the following format:

   ```
   sqlplus {username | /} [as sysdba]
   ```

   An example of this command is:

   ```
   $ sqlplus / AS SYSDBA
   Enter password: password
   ```

   For *username*, you can use the SYS or SYSTEM administrative users. At the prompt, enter the password that you set up during installation. If you use the SYS user, you must include AS SYSDBA after the username.

   SQL*Plus connects you to the default database instance (Microsoft Windows) or the database instance specified by environment variables (Linux and UNIX).

To start SQL*Plus and connect to the database from the Windows Start menu:

1. Configure the operating system environment variables, as described in "Configuring the Operating System Environment Variables ."
2. Click **Start**, select **Programs** (or **All Programs**), then **Oracle - *HOME_NAME***, then **Application Development**, and then **SQL*Plus**.
3. When prompted, enter the **user name** and **password** for the account to use to connect to the database.

   For the user name, you can use the SYS or SYSTEM administrative accounts, and you can use the password that you set up during installation.

   If you use SYS or / as the user name, follow them with a space and then the clause AS SYSDBA, as shown in the following examples:

   ```
   Enter user-name: SYS AS SYSDBA
   Enter password: password

   or

   Enter user-name: / AS SYSDBA
   ```

See Also:

- "SYS and SYSTEM Users" for information about the recommended alternative to using the `SYSTEM` account for day-to-day administrative tasks
- "Connecting to an Oracle Database from a Client Computer"
- "About Administrative Accounts and Privileges"
- *Oracle Database Administrator's Guide*
- *SQL*Plus User's Guide and Reference*

About SQL Developer

SQL Developer provides another GUI for accessing your Oracle database. SQL Developer supports development in both the SQL and PL/SQL languages. It is available in the default installation of Oracle Database.

With SQL Developer, you can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. You can also run any number of provided reports, and also create and save your own.

You can also download the latest release of SQL Developer from the Oracle Technology Network (OTN) Web site.

See Also:

- "About SQL Developer" for information about installing and using SQL Developer
- *Oracle Database 2 Day Developer's Guide* for instructions for starting SQL Developer
- *Oracle Database Concepts*

Getting Started with Oracle Enterprise Manager: Oracle by Example Series

Oracle by Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter and includes annotated screenshots.

To view the Getting Started with Oracle Enterprise Manager Express OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6283,1
```

4

Configuring the Network Environment

After installing Oracle Database, you have a fully functional database with a client/server network environment that has been minimally configured.

This chapter contains the following sections to help you completely configure your client/server network:

- Understanding Network Configuration
- Viewing Listener Configuration
- Starting and Stopping the Listener
- Connecting to an Oracle Database from a Client Computer
- Networking: Oracle by Example Series

Understanding Network Configuration

A **client** is any application that connects to Oracle Database to send or retrieve data. An Oracle Database client application can reside on any computer that has Oracle Database client software installed.

Oracle Net is a software layer that resides on the client computer and on the Oracle Database host computer. It establishes and maintains the connection between the client application and the database over a network, and exchanges messages between them using industry standard protocols.

For a client application and a database to communicate, the client application must be able to identify the database it wants to connect to, and the database must provide an identification. You can use a service name to connect to a database. A **service name** is a logical representation of a database, which is the way a database is presented to clients. A single database can be presented as multiple services.

Service names can provide location transparency so that the client application does not have to know the server's location. If the database is moved to another location, then you must reconfigure only Oracle Net. No changes are necessary to client applications.

This section contains these topics:

- Oracle Net Listener Configuration
- Client Connections
- Tools for Network Configuration

Oracle Net Listener Configuration

On the database host, the Oracle Net listener (the listener), is a process that listens for client connection requests. It receives incoming client connection requests and manages the traffic of these requests to the database server.

The default listener configuration file is called `listener.ora`, and it is located in the `network/admin` subdirectory of the Oracle home directory. For example, if your Oracle home directory is `/u01/app/oracle/product/11.2.0/dbhome_1`, then the `listener.ora` file is created by default in the `/u01/app/oracle/product/11.2.0/dbhome_1/network/admin` directory.

The file contains a protocol address that identifies the database. This address defines the protocol the listener is listening on and any other protocol-specific information. For example, the listener could be configured to listen at the following protocol address:

```
(DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=dbhost.example.com) (PORT=1521)))
```

This example shows a TCP/IP protocol address that specifies the host computer of the listener and a port number. The listener can listen for connection requests on network interfaces with either IP version 4 (IPv4) or IP version 6 (IPv6) addresses.

The `listener.ora` file is automatically configured during installation.

Because the configuration parameters have default values, you can start and use a listener without configuring it. This default listener is named `LISTENER`, supports no service names on startup, and listens on the following TCP/IP protocol address:

```
(ADDRESS=(PROTOCOL=tcp)(HOST=host_name)(PORT=1521))
```

An Oracle database registers with the listener within a minute or so of starting up. The service names, or the databases that they represent, to which the listener forwards client requests, can be configured in the `listener.ora` file. This information can also be dynamically registered with the listener. Dynamic registration of services and databases with the listener is called **service registration**.

Service registration is performed by the listener registration (LREG) process—an instance background process—of each database instance. Dynamic service registration does not require modification of the `listener.ora` file.

See Also:

- *Oracle Database Concepts* for more information about listeners and service names
- *Oracle Database Net Services Administrator's Guide* for more information about configuring listeners
- "Viewing Listener Configuration"
- "Starting and Stopping the Listener"
- "About Background Processes" for more information about database processes

---

Client Connections

The following sections describe the elements involved in client connections to the database:

- Connect Descriptors
- Connection Requests
- Naming Methods

Connect Descriptors

The client uses a connect descriptor to specify the database to which it wants to connect. This connect descriptor contains a protocol and a database service name. A database can have multiple service names defined, so a specific service name must be specified for the connect descriptor. In a preconfigured database, there is only one service name, which defaults to the global database name.

The following example shows a connect descriptor that enables clients to connect to a database with service name `mydb.us.example.com`:

```
DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=my-server) PORT=1521))
   (CONNECT_DATA=
      (SERVICE_NAME=mydb.us.example.com))
```

Connection Requests

Users initiate a connection request by providing a connect string. A connect string includes a user name and password, and a connect identifier. This connect identifier can be the connect descriptor itself, or a name that resolves to the connect descriptor using mapping information stored in one or more repositories accessed with the naming methods described in "Naming Methods". This name is referred to as a **net service name**.

Naming Methods

A **naming method** is a resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service.

Oracle Net provides support for the following naming methods:

- **Easy Connect Naming**

  The easy connect naming method enables clients to connect to an Oracle database by using only a TCP/IP connect string consisting of a host name and service name. The easy connect naming method requires no configuration. See "Connecting to an Oracle Database from a Client Computer" for an example of easy connect naming.

- **Local Naming**

  The local naming method stores connect descriptors, identified by their net service names, in a client configuration file named `tnsnames.ora`. This file is located in the *`ORACLE_HOME`*`/network/admin` directory. When you create a database using Oracle Database Configuration Assistant (DBCA), local naming is configured automatically. You must then use the Net Configuration Assistant to create connect descriptors and their corresponding net service names.

- **Directory Naming**

Directory naming resolves a database service, net service name, or net service alias to a connect descriptor stored in an LDAP-compliant directory server.

See Also:

- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Administrator's Guide* for more information about database services

Tools for Network Configuration

Oracle Database enables you to manage your network configuration with the following tools:

- Net Configuration Assistant
- Oracle Net Manager

Net Configuration Assistant

During a typical database installation, Net Configuration Assistant automatically configures a listener called `LISTENER` that has a TCP/IP listening protocol address for the database. If you do a custom installation, then Net Configuration Assistant prompts you to configure a listener name and protocol address of your choice.

Use Net Configuration Assistant for initial network configuration after database installation. Thereafter, you can use Oracle Net Manager to configure and administer your networks.

Oracle Net Manager

Oracle Net Manager provides various network configuration features, including the ability to configure profiles.

Note:

On Microsoft Windows operating systems only, if a non-administrator, low privileged Windows User Account was specified as the Oracle Home User during database installation, you are prompted for the Oracle Home User password when you access the Net Configuration Assistant and Oracle Net Manager.

See Also:

- "Viewing Listener Configuration"
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Platform Guide for Microsoft Windows* for more information about the Oracle Home Users on Windows feature

Viewing Listener Configuration

The Oracle Net listener (the listener) runs on your database host and handles incoming client requests. You can view the listener status at the command line.

To view information about the listener at the command line:

1. Open a command window.
2. Follow the steps listed in "Configuring the Operating System Environment Variables ."
3. Enter the following command:

```
lsnrctl status
```

Starting and Stopping the Listener

The Oracle listener is set to start automatically whenever the host is restarted. However, when your system encounters unforeseen circumstances, or when you have manually stopped the listener, you can restart it at the command line.

To start or stop the listener at the command line:

1. Open a command window.
2. Follow the steps listed in "Configuring the Operating System Environment Variables ."
3. Enter either of the following commands, depending on whether you want to start or stop the listener:

```
lsnrctl start
lsnrctl stop
```

Connecting to an Oracle Database from a Client Computer

This section describes how to use SQL*Plus and the easy connect naming method to connect to an Oracle database from a client computer. SQL*Plus is typically installed when you install Oracle Database client software. The easy connect naming method provides TCP/IP connectivity to databases without requiring you to configure Oracle Net Services.

You can use the instructions in this section to test your network configuration.

To connect to an Oracle database from a client computer using easy connect naming:

1. Complete the steps in "Configuring the Operating System Environment Variables ".
2. Do one of the following to start SQL*Plus:
   - (UNIX, Linux, or Windows systems) Open a command window and enter the following command:

     ```
     sqlplus
     ```

- (Windows systems only) Click **Start**, select **Programs** (or **All Programs**), then **Oracle - *HOME_NAME***, then **Application Development**, and then **SQL*Plus**.

3. When prompted, enter the user name followed by an at sign (@) and a connect identifier, where the connect identifier has the following format:

   ```
   "host[:port][/service_name][:server][/instance_name]"
   ```

   The place holders used in the connect identifier format represent:

   - *host* — the name or IP address of the Oracle database host computer.

     Both IPv4 and IPv6 addresses are supported. IPv6 addresses must be enclosed in square brackets. See *Oracle Database Net Services Administrator's Guide* for information about IPv6 addressing.

   - *port* (optional) — the TCP port number on which the Oracle Net listener listens for connections.

     If *port* is excluded, then the standard port number 1521 is assumed.

   - *service_name* — a database service name.

     If no database service names are defined, then you can use the name of the default service that is created for the database. This service name consists of the global database name, which is made up of the DB_NAME and DB_DOMAIN initialization parameters as follows:

     ```
     DB_NAME.DB_DOMAIN
     ```

     If DB_DOMAIN is null, then the standard service name is just DB_NAME.

   - *server* — the type of service handler. Acceptable values are dedicated, shared, and pooled. If omitted, then the default type of server is chosen by the listener: shared server if configured, otherwise dedicated server.

   - *instance_name* — the instance to which to connect. When you specify only instance name, you connect to the default database service. If there is no default service configured in the listener.ora file, then an error is generated.You can obtain the instance name from the INSTANCE_NAME initialization parameter.

   For example, to connect as user NICK to the database service orcl.example.com on the host dbhost.example.com, enter the following at the user name prompt:

   ```
   nick@"dbhost.example.com/orcl.example.com"
   ```

   The following examples substitute IPv4 and IPv6 addresses for the host name:

   ```
   nick@"192.0.2.1/orcl.example.com"
   nick@"[2001:0DB8:0:0::200C:417A]/orcl.example.com"
   ```

4. When prompted, enter the user password.

See Also:

- *Oracle Database Administrator's Guide* for more examples of connecting with SQL*Plus and for a discussion of environment variables
- *Oracle Database Net Services Administrator's Guide* for information about easy connect, connect identifiers, and other naming methods
- *Oracle Database Net Services Reference* for information about how to define the default service in `listener.ora`
- *SQL*Plus User's Guide and Reference*

Networking: Oracle by Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter and includes annotated screenshots.

To view the Using the Listener Control Utility to Manage the Listener OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6284,1
```

5

Managing the Oracle Instance

This chapter provides background information about the Oracle instance and instructions for managing the instance.

This chapter contains the following sections:

Overview of the Oracle Instance and Instance Management

An Oracle database system consists of an Oracle database and an Oracle instance (in an Oracle Real Application Clusters environment, there can be more than one instance).

A **database** consists of a set of disk files that store user data and metadata. **Metadata**, or "data about the data," consists of structural, configuration, and control information about the database.

An **Oracle instance** (also known as a **database instance**) contains the set of Oracle Database background processes that operate on the stored data and the shared allocated memory that those processes use to do their work.

Each instance has an instance ID, also known as a system ID (SID). Because there can be multiple Oracle databases on a host computer, each with its own set of data files, you must identify the instance to which you want to connect. For a local connection, you identify the instance by setting operating system environment variables `ORACLE_SID` and `ORACLE_HOME`. For a remote connection, you identify the instance by specifying a network address and a database service name.

An Oracle instance must be started to read and write information to the database. The Oracle instance creates the database upon receipt of instructions from the Oracle Database Configuration Assistant (DBCA) utility or the `CREATE DATABASE` SQL statement.

When the Oracle instance is not available, your data is safe in the database, but it cannot be accessed by any user or application.

The properties of an Oracle instance are specified using instance initialization parameters. When the instance is started, an initialization parameter file is read, and the instance is configured accordingly.

This section presents some concepts of an Oracle instance and its management. It contains the following topics:

- About Initialization Parameters
- About Background Processes
- About Server and Client Processes
- About Instance Memory Structures
- About Instance Startup and Shutdown

See Also:

Oracle Database Concepts for an overview of the Oracle database instance

About Initialization Parameters

Managing an Oracle instance includes configuring parameters that affect the basic operation of the Oracle instance. These parameters are called initialization parameters. The Oracle instance reads initialization parameters from a file at startup.

During installation, when you select a preconfigured database workload available in Database Configuration Assistant (DBCA), the initialization parameters are optimized for typical use in the environment that you specified. As the number of database users increases and the workload increases, you might have to alter some initialization parameters. You can make these changes using the Initialization Parameter page in Oracle Enterprise Manager Database Express (EM Express), or by using an advisor provided by Oracle Database, such as the Memory Advisor. See "Optimizing Memory Usage with the Memory Advisors" for more information.

After being read from a file, initialization parameters are retained in memory, where the values for many of them can be changed dynamically. There are two types of parameter files. The type of file used to start the instance determines if dynamic initialization parameter changes persist across database shutdown and startup. The parameter file types are:

- Server parameter file

  The **server parameter file**, commonly known as the SPFILE, is the preferred form of initialization parameter file, and is a binary file that can be written to and read by the database. It *must not* be edited manually. It is stored on the host computer on which Oracle Database is running. Changes are made when you use EM Express or SQL*Plus to modify one or more initialization parameters, or when Oracle Database itself makes changes for self-tuning purposes. Any changes to it persist across database shutdown and startup operations.

  Note:

  When changing an initialization parameter in the server parameter file, you can also specify that the in-memory value be changed, so that your change is reflected immediately in the current instance. If you do not change the in-memory value, then the change does not take effect until you shut down and restart the database.

- Text initialization parameter file

  A **text initialization parameter file** is a text file that can be read by the Oracle instance, but it is not written to by the instance. You can change a text initialization parameter file with a text editor, but changes do not take effect until you restart the Oracle instance. When you start the instance with this type of file, you can still change many initialization parameters dynamically with EM Express, but only for the current instance. Unless you also edit the text initialization parameter file and make the same change, the change is lost when you restart the database instance.

You can use SQL statements to create the following:

- A server parameter file from a text initialization file
- A server parameter file from the current (in-memory) values of all initialization parameters
- A text initialization parameter file from a server parameter file

When you create the database with DBCA, a server parameter file is created. This file is then used each time the database is started.

See Also:

- "Viewing and Modifying Initialization Parameters"
- *Oracle Database Administrator's Guide* for information about the default name and location of the server parameter file, and for commands to create a server parameter file or text initialization parameter file
- *Oracle Database Concepts* for an overview of parameter files

About Background Processes

The background processes of the Oracle instance manage memory structures, asynchronously perform I/O to write data to a file on a disk, and perform general maintenance tasks. The background processes consolidate functions that would otherwise be handled by multiple Oracle Database programs running for each user process. They monitor other Oracle Database processes to provide increased parallelism for better performance and reliability.

The background processes that are present depend on the features that are being used in the database. Some fundamental background processes are described in Table 1 .

Table 1 Oracle Database Background Processes

Background Process	Description
Database writer (DBWn)	The database writer writes modified blocks from the database buffer cache to the files on a disk. Oracle Database allows a maximum of 36 database writer processes.
Log writer (LGWR)	The log writer process writes redo log entries to disk. Redo log entries are generated in the redo log buffer of the System Global Area (SGA) and the log writer process writes the redo log entries sequentially into an online redo log file.
Checkpoint (CKPT)	At specific times, all modified database buffers in the SGA are written to the data files by a database writer process (DBW*n*).

Background Process	Description
	This event is called a **checkpoint**. The checkpoint process signals DBW*n*, updates the data files and control files of the database, and records the time of this update.
System monitor (SMON)	The system monitor performs instance recovery when a failed instance is restarted.
Process monitor (PMON)	The process monitor performs a recovery when a user process fails. It cleans up the cache and frees resources that the failed process was using.
Archiver (ARCn)	Archiver processes copy the online redo log files to archival storage when the log files are full or a log switch occurs. The database must be in archive log mode to run archive processes. For more information, see Performing Backup and Recovery.
Manageability monitor (MMON)	This process performs various management-related background tasks, for example: • Issuing alerts whenever a given metric violates its threshold value • Taking snapshots by spawning additional processes • Capturing statistical values for SQL objects that have been recently modified
Job Queue Processes (CJQ0 and Jnnn)	Job queue processes run user jobs, often in batch mode. A **job** is a user-defined task scheduled to run one or more times.

See Also:

- *Oracle Database Reference* for a complete list of Oracle Database background processes
- *Oracle Database Concepts* for an overview of background processes

About Server and Client Processes

In addition to background processes, Oracle Database creates server processes that handle the connection requests of user or client processes. A user connection is composed of the following distinct pieces:

- A client program acting on behalf of the user, such as Oracle Enterprise Manager (Enterprise Manager), SQL*Plus, or an application
- A server process that handles the connection to the database on behalf of the client program, and that performs much of the work for the client program, such as parsing and running SQL statements, and retrieving and returning results to the client program

Server processes can be either dedicated or shared. When server processes are dedicated, Oracle Database is running in **dedicated server mode**. When server processes are shared, Oracle Database is running in **shared server mode**. In dedicated server mode, each client process has its own server process. Although a dedicated server process is good for long-running queries and administrative tasks, an idle process or too many dedicated processes can result in an inefficient use of resources.

Using shared server mode eliminates the need for a dedicated server process for each user connection, requires less memory for each user connection, and enables more users to access the database. Shared server mode is more efficient at supporting multiple client programs making frequent short-running queries.

About Instance Memory Structures

The sizes of the instance memory structures affect database performance and are controlled by initialization parameters.

Upon installation, you can let the database manage memory for you automatically, or you can choose to manually configure the instance memory structures. If you choose manual memory management, then Oracle Database provides advisors to help you determine appropriate values for memory parameters.

The following sections describe the two important memory areas that you can monitor and size:

- System Global Area
- Program Global Area

See Also:

- "Managing Memory"
- *Oracle Database Concepts* and *Oracle Database Administrator's Guide* for more information about Oracle Database memory architecture

System Global Area

The System Global Area (SGA) is a group of shared memory structures, known as SGA components, that contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.

Table 2 lists the components of the SGA.

Table 2 SGA Components

Component	Description
Database buffer cache	Before data stored in the database can be queried or modified, it must be read from a disk and stored in the buffer cache. All user processes connected to the database share access to the buffer cache. For optimal performance, the buffer cache should be large enough to avoid frequent disk I/O operations.
Shared pool	The shared pool caches information that is shared among users: • SQL statements that can be reused • Information from the data dictionary such as user account data, table and index descriptions, and privileges • Stored procedures, which are executable code that is stored in the database
Redo log buffer	This buffer improves performance by caching redo information until it can be written to the physical online redo log files stored on disk. Redo information and online redo log files are discussed in "About Online Redo Log Files".

Component	Description
Large pool	This optional area is used to buffer large I/O requests for various server processes.
Java pool	The Java pool is an area of memory that is used for all session-specific Java code and data within the Java Virtual Machine (JVM).
Streams pool	The Streams pool is an area of memory that is used by the Oracle Streams feature. For more information about Oracle Streams, see *Oracle Streams Concepts and Administration*.
Result cache	The result cache buffers query results. If a query is run for which the results are stored in the result cache, then the database returns the query results from the result cache instead of rerunning the query. This SGA component speeds the execution of frequently run queries.

See Also:

- "About Instance Memory Structures"
- *Oracle Database Concepts* for more information about the SGA

Program Global Area

A **Program Global Area (PGA)** is a memory region that contains data and control information for a server process. It is nonshared memory created by Oracle Database when a server process is started. Access to the PGA is exclusive to the server process. There is one PGA for each server process. Background processes also allocate their own PGAs. The total PGA memory allocated for all background and server processes attached to an Oracle Database instance is referred to as the **total instance PGA memory**, and the collection of all individual PGAs is referred to as the **total instance PGA**, or just **instance PGA**.

The amount of PGA memory used and the contents of the PGA depend on whether the instance is running in dedicated server or shared server mode.

The PGA is used to process SQL statements and to hold logon and other session information. A large part of the PGA is dedicated to **SQL work areas**, which are working memory areas for sorts and other SQL operations.

See Also:

- "About Server and Client Processes" for more information about dedicated server and shared server modes
- *Oracle Database Concepts* for more information about the PGA

About Instance Startup and Shutdown

After installation, the Oracle instance is started, and the database is open for access by users with database accounts. At some point, you may want to shut down and restart the instance. This section describes the startup and shutdown processes.

The phrases "starting up and shutting down the Oracle instance" are often used interchangeably with "starting up and shutting down the database."

This section contains the following topics:

- About Administration Privileges for Startup and Shutdown
- About Instance Startup
- About Instance Shutdown

See Also:

- "Shutting Down and Starting Up the Oracle Instance"
- *Oracle Database Concepts*

About Administration Privileges for Startup and Shutdown

To start or shut down the Oracle instance, you must connect to the instance with a special connection privilege. There are two of these privileges: SYSDBA for fully empowered database administrators and SYSOPER for users who start and shut down the database, but have no privileges to access user objects.

When you create an Oracle database, there are two primary administrative user accounts that are automatically created: SYS and SYSTEM. Both of these users have full database administration privileges, but initially, only user SYS or SYSTEM can connect with the SYSOPER privilege. Therefore, until you grant the SYSOPER privilege to other users, you must connect to the Oracle instance as user SYS or SYSTEM to start and shut down the instance. When connecting (logging in) as user SYS, you must always specify that you are connecting AS SYSDBA.

See Also:

- "About Administrative Accounts and Privileges" for more information about the SYS and SYSTEM users, and the SYSDBA privilege
- "SYS and SYSTEM Users" for information about the recommended alternative to using the SYSTEM account for day-to-day administrative tasks

About Instance Startup

When you start the Oracle instance, you typically start it such that the state of the database is OPEN and ready for user connections. However, there are situations in which you may want to start the instance with the database in the MOUNTED state, but not open. An instance can also be started without the database either mounted or open. Thus, there are three stages to starting an instance:

1. You start the instance using one of the following methods:
   - Using the SQL*Plus STARTUP command. See "Shutting Down and Starting Up Using SQL*Plus".

- On Microsoft Windows, using the Services program in Control Panel to start the Oracle Database services. See "Shutting Down and Starting Up Using the Windows Services Program".

The instance reads the initialization parameter file, allocates System Global Area (SGA) memory, and starts the background processes.

2. If you mount the database, then the Oracle instance opens the control file for the database, but does not open the data files. The database is now considered to be in the `MOUNT` state. This state enables you to perform certain administrative functions that cannot be performed when other users are accessing the database. An example of such a function is enabling or disabling the archiving of online redo log files. See "About Archived Redo Log Files" for information about online redo log file archiving.

3. If you open the database, then, after reading the parameter file and control file, the online redo log files and data files for the database are also opened. The state of the database is now `OPEN` and user access to the data is available.

The default startup mode for the database (OPEN) completes the three stages in sequence. Unless you explicitly specify otherwise, the instance is started, the database is mounted, and then the database is opened.

About Instance Shutdown

Instance shutdown is the reverse of instance startup. When you shut down the Oracle instance, the default mode is a `NORMAL` shutdown, which means users are not allowed to create new connections to the database, but the shutdown process waits for all currently connected users to exit their sessions. After all the users have disconnected, then the committed transactions are written to disk, the database files are closed, and the instance is stopped. However, there are situations in which you may not want to wait for users to disconnect on their own (`IMMEDIATE` mode), or you want to let the current transactions for each user complete before they are disconnected (`TRANSACTIONAL` mode). In emergency situations you can even shut down the database without waiting for the committed transactions to be written to disk (`ABORT` mode).

Shutting down an instance goes through the following stages:

1. After all the users have exited from their sessions, or been disconnected, Oracle Database writes data in the System Global Area (SGA) to the data files and online redo log files. A checkpoint is performed on the data files and their headers are marked current as of the time of the instance shutdown. The data files and online redo log files are then closed and the state of the database is changed to `CLOSED`. The control file remains open to the instance.

2. The Oracle instance dismounts the database and updates relevant entries in the control file to record a clean shutdown. The control file is closed. The database is now closed and dismounted. The instance is in the `NOMOUNT` state.

3. The Oracle instance stops the background processes and deallocates the shared memory used by the SGA.

If a `SHUTDOWN ABORT` or abnormal termination occurs, then the instance of an open database closes and shuts down the database instantaneously. Oracle Database does not write data in the buffers of the SGA to the data files and redo log files. The subsequent reopening of the database requires instance recovery, which Oracle Database performs automatically.

Shutting Down and Starting Up the Oracle Instance

This section provides instructions about two methods you can use to start or shut down the Oracle instance:

- Shutting Down and Starting Up Using SQL*Plus
- Shutting Down and Starting Up Using the Windows Services Program

See Also:

"About Instance Startup and Shutdown"

Shutting Down and Starting Up Using SQL*Plus

You can shut down and start the Oracle instance using SQL*Plus.

To shut down and start the Oracle instance using SQL*Plus:

1. Start SQL*Plus and connect to the database.

   See "Shutting Down and Starting Up Using SQL*Plus".

2. Issue the SQL*Plus `SHUTDOWN` command:

   ```
   SQL> SHUTDOWN
   ```

   The database instance is shut down.

   The `NORMAL` clause of the `SHUTDOWN` command is optional because this is the default shutdown method.

   See "About Instance Shutdown" for more information about when to use other `SHUTDOWN` command options.

3. Restart the database by issuing the SQL*Plus `STARTUP` command:

   ```
   SQL> STARTUP
   ```

   The database instance is restarted.

   See "About Instance Startup" for more information about when to use other `STARTUP` command options.

Shutting Down and Starting Up Using the Windows Services Program

On Microsoft Windows, you can also start and shut down your Oracle database using the Services program in Control Panel. You must start or stop the following services:

- `OracleService`*`SID`*, which is your Oracle instance.
- `Oracle`*`ORACLE_HOME`*`TNSListener`, which is your listener. The listener is required for clients to connect to your database.

In the preceding service names, *`SID`* refers to the system identifier for the instance and *`ORACLE_HOME`* refers to the Oracle home name.

To start or stop Oracle Database services:

1. Click **Start** and then select **Control Panel**.

   The Control Panel window opens.

2. Double-click the **Administrative Tools** icon, and then double-click the **Services** icon.

   The Services window opens, displaying all Windows services that are available on your system.

3. Locate the Oracle Database services listed at the beginning of this section. For example, if your SID is `orcl`, then locate the following services:

   - `OracleServiceORCL`
   - `OracleOraDb11g_home1TNSListener`

4. Start or stop the services, using the following steps for each service:

   a. Select the service name.

   b. In the Action menu, click **Start** or **Stop**.

Viewing and Modifying Initialization Parameters

This section provides instructions about viewing the initialization parameter settings for your database and modifying these parameters. You can modify the initialization parameters for the database in one of three ways:

- Until the instance is shut down: The new values for the initialization parameters are applied to the currently running instance, but, when the database is restarted, the initialization parameter values revert to their previous settings.
- From now until the initialization parameter is changed again: The changes are applied to the currently running instance and are also stored in the server parameter file. The changes made to the initialization parameters persist when the database is restarted.
- When the database is restarted: The new values for the initialization parameters are recorded in server parameter file, but are not applied to the currently running instance. The changes take effect only when the database is restarted.

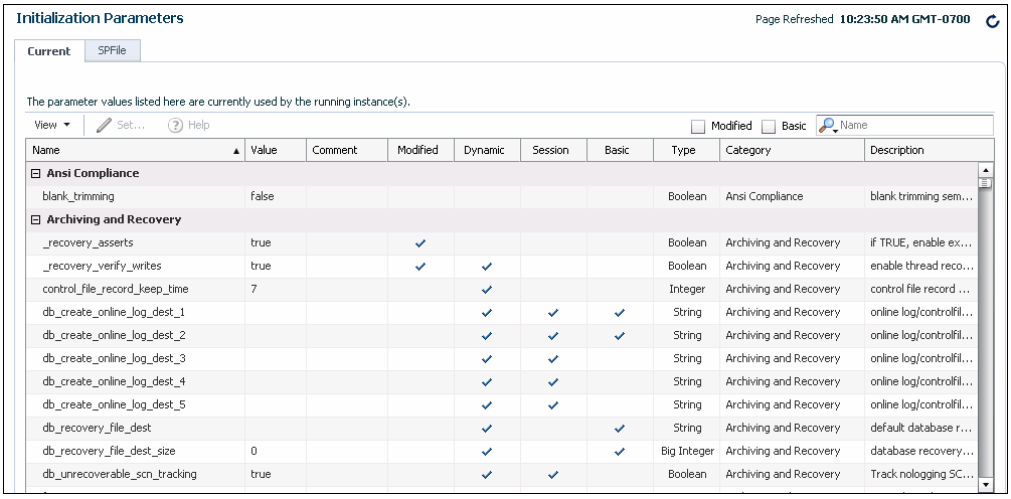
Note:

These three scenarios correspond to using the `SCOPE=MEMORY`, `SCOPE=BOTH`, and `SCOPE=SPFILE` clauses of the `ALTER SYSTEM` SQL statement, respectively, when you use the `ALTER SYSTEM` statement to change initialization parameters.

To view or modify initialization parameters:

1. In Oracle Enterprise Manager Database Express (EM Express), from the **Configuration** menu, select **Initialization Parameters**.

   The Initialization Parameters page appears.



The Initialization Parameters page has two tabs:

- **Current**—This tab (the default) displays all initialization parameter values that are currently active (in memory) for the Oracle instance.
- **SPFile**—This tab displays initialization parameter settings in the server parameter file (SPFILE). This tab is present only when the current instance started up with a server parameter file. The file location is displayed at the top of the tab.

Note:

In a pluggable database (PDB), the Initialization Parameters page includes the PDB Modifiable column. Each initialization parameter that can be modified at the PDB level has a check mark in the PDB Modifiable column.

Any initialization parameter in a PDB that does not have a check mark in the PDB Modifiable column can be set and modified only in the root, and the value set in the root applies to the individual PDBs in the multitenant container database (CDB).

For more information on individual initialization parameters, see *Oracle Database Reference*.

2. (Optional) On either tab, reduce the number of initialization parameters displayed by doing one or both of the following:
   - In the search field, enter text.
   - Select either the **Modified** or **Basic** option next to the search field to limit the display to either modified or basic initialization parameters.

   For example, to view only initialization parameters that have the text `DEST` anywhere in the parameter name, enter `dest` in the Search field. EM Express then restricts the list of initialization parameters accordingly.

3. To modify an initialization parameter for the currently running instance only (the modifications will not persist when the instance is restarted), complete the following steps:

a. On the Current tab, select the initialization parameter whose value you want to modify.

Note:

If the **Set** button does not become available when you select the parameter, then the parameter is not dynamic—that is, it cannot be changed for the current instance.

---

b. Click the **Set** button.

The Set Initialization Parameter page appears.

c. In the **Value** column, enter a new value for the initialization parameter.

d. For the **Scope** field, ensure that **Memory** is selected. The value you set will not persist when the instance is restarted.

e. (Optional) In the **Comments** column, enter text explaining the reasons for the changes.

f. Click **OK**.

A confirmation message appears.

4. To modify an initialization parameter for the currently running instance, and also record the modifications in the server parameter file that will persist when the database instance is restarted, complete the following steps:

a. On the Current tab, select the initialization parameter whose value you want to modify.

Note:

If the **Set** button does not become available when you select the parameter, then the parameter is not dynamic—that is, it cannot be changed for the current instance.

---

b. Click the **Set** button.

The Set Initialization Parameter page appears.

c. For the **Scope** field, ensure that both **Memory** and **SPFile** are selected. The value you set will persist when the database instance is restarted.

Note:

If the **SPFile** option is not available, then the database instance does not have an SPFILE, and changes made to the instance will not persist when the instance is restarted.

---

d. (Optional) In the **Comments** column, enter text explaining the reasons for the changes.

e. Click **OK**.

A confirmation message appears. The message includes a Show SQL button. Click the Show SQL button to see the SQL statement that was executed.

5. To modify an initialization parameter in the server parameter file only, such that the current instance is not affected and changes take effect only when the database is next restarted, complete the following steps:

   a. Click **SPFile** to view the SPFile tab.

   b. Select the initialization parameter whose value you want to modify. If the initialization parameter does not appear on the SPFile tab, then select the initialization parameter on the Current tab, instead.

   c. Click the **Set** button.

      The Set Initialization Parameter page appears.

   d. In the **Scope** field, choose **SPFile**.

      For an initialization parameter that cannot be reset without restarting the database, the **Scope** field defaults to **SPFile**, and the **Memory** option does not appear.

   e. In the **Value** column, enter a new value for the initialization parameter.

   f. (Optional) In the **Comments** column, enter text explaining the reasons for the changes.

   g. Click **Apply**.

      A confirmation message appears.

Note:

Changes to initialization parameters are recorded in the alert log as `ALTER SYSTEM` statements. See *Oracle Database Administrator's Guide* for information about the alert log.

See Also:

"About Initialization Parameters"

Managing Memory

This section provides background information about managing memory for the Oracle instance, and includes instructions about how to adjust the memory allocation for the Oracle instance. It contains the following topics:

- About Memory Management
- Enabling Automatic Memory Management
- Modifying Memory Settings – Automatic Memory Management
- Enabling Automatic Shared Memory Management
- Modifying Memory Settings – Automatic Shared Memory Management

- Enabling Manual Shared Memory Management
- Modifying Memory Settings - Manual Shared Memory Management
- Modifying Memory Settings – Automatic PGA Memory Management

See Also:

"About Instance Memory Structures"

About Memory Management

Memory management involves maintaining optimal sizes for the Oracle instance memory structures as demands on the database change. The memory that must be managed is the System Global Area (SGA) memory and the instance Program Global Area (PGA) memory. The instance PGA memory is the collection of memory allocations for all individual PGAs.

Beginning with Oracle Database 11*g* Release 1 (11.1), you can let the database manage the SGA memory and instance PGA memory completely. You designate only the total memory size to be used by the instance, and Oracle Database dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands. This capability is referred to as **automatic memory management**. In this memory management mode, the database also dynamically tunes the sizes of the individual SGA components and the instance PGA.

To have more direct control over the sizes of the SGA and instance PGA, you can disable automatic memory management and enable automatic shared memory management.

- With **automatic shared memory management**, you set target and maximum sizes for the SGA. Oracle Database then tunes the total size of the SGA to your designated target, and dynamically tunes the sizes of all SGA components.
- When you enable automatic shared memory management, you can also enable automatic PGA memory management. With **automatic PGA memory management**, Oracle Database automatically performs memory management of instance PGA. Optionally, you can set a target size for the instance PGA, and the database then tunes the size of the instance PGA to your target, and dynamically tunes the sizes of individual PGAs.

If you want complete control of individual SGA component sizes, you can disable both automatic memory management and automatic shared memory management. This is called **manual shared memory management**. In this mode, you set the sizes of several individual SGA components, thereby determining the overall SGA size. You then manually tune these individual SGA components on an ongoing basis.

Manual shared memory management mode is intended for experienced DBAs only. Note that in this mode, automatic PGA memory management remains enabled.

Note:

Although it is possible to disable automatic PGA memory management, it is not recommended, and is not described in this manual.

Table 3 summarizes the various memory management modes that you can set for your database instance.

Table 3 Oracle Database Memory Management Modes

Memory Management Mode	You Set	Oracle Database Automatically Tunes
Automatic memory management	• Total memory size for this instance • (Optional) Maximum memory size for this instance	• Total SGA size • SGA component sizes • Instance PGA size
Automatic shared memory management and automatic PGA memory management (Automatic memory management disabled)	• SGA target size • (Optional) SGA maximum size • (Optional) Instance PGA target size	• SGA component sizes
Manual shared memory management and automatic PGA memory management (Automatic memory management and automatic shared memory management disabled)	• Shared pool size • Buffer cache size • Java pool size • Large pool size • (Optional) Instance PGA target size	• Instance PGA size

Note:

Automatic Memory Management is not available on all platforms. See *Oracle Database Administrator's Guide* for more information about supported platforms.

If you choose the basic installation option when you install the database, then automatic memory management is enabled. If you choose advanced installation, then Database Configuration Assistant (DBCA) enables you to select from the three memory management modes. Oracle recommends that you enable automatic memory management.

Whichever memory management mode you choose, you may have occasion to adjust memory settings as demands on the database or on its host computer change. Reasons why you adjust memory settings include the following:

- You receive a memory-related alert or error message.
- You receive a memory-related recommendation from Automatic Database Diagnostic Monitor (ADDM).
- You want to change the amount of memory allocated to accommodate future growth in memory demand.

You can use a *memory advisor* to help you adjust memory sizes. See "Modifying Memory Settings – Automatic Shared Memory Management" for an example of using a memory advisor.

Note:

The initialization parameters that are used to manage memory are set in the root of a multitenant container database (CDB), and the values set for those parameters are applied to all of the pluggable databases (PDBs) in the CDB.

See Also:

- "Enabling Automatic Memory Management"
- "Modifying Memory Settings – Automatic Memory Management"
- "Modifying Memory Settings – Automatic PGA Memory Management"

Enabling Automatic Memory Management

If you did not enable automatic memory management when you installed and configured your database, then Oracle recommends that you do so after installation, unless you are an experienced DBA with specific reasons to manually tune memory sizes. With automatic memory management, the Oracle instance dynamically tunes all memory components to optimize performance as the workload changes.

To enable automatic memory management:

1. Start SQL*Plus and connect to the database as SYSDBA.
2. Calculate the minimum value for MEMORY_TARGET as follows:
   a. Determine the current sizes of SGA_TARGET and PGA_AGGREGATE_TARGET by entering the following SQL*Plus command:

```
SHOW PARAMETER TARGET
```

SQL*Plus displays the values of all initialization parameters with the string TARGET in the parameter name.

```
NAME                           TYPE        VALUE
------------------------------ ----------- ----------------
archive_lag_target             integer     0
db_flashback_retention_target  integer     1440
fast_start_io_target           integer     0
fast_start_mttr_target         integer     0
memory_max_target              big integer 0
memory_target                  big integer 0
parallel_servers_target        integer     32
pga_aggregate_target           big integer 29M
sga_target                     big integer 356M
```

Or, on the Initialization Parameters page in Oracle Enterprise Manager Database Express (EM Express), you can enter "TARGET" in the Search field to display the values of all the initialization parameters with the string TARGET in the parameter name, as described in "Viewing and Modifying Initialization Parameters."

   b. Run the following query to determine the maximum instance Program Global Area (PGA) allocated since the database was started:

```
SQL> select value from v$pgastat where name='maximum PGA allocated';

      VALUE
----------
 246844416
```

246844416 bytes is approximately 235M.

c. Compute the maximum value between the query result from step 2.b and PGA_AGGREGATE_TARGET. Add SGA_TARGET to this value.

```
memory_target = sga_target + max(pga_aggregate_target, maximum PGA
allocated)
```

For example, if SGA_TARGET is 356M and PGA_AGGREGATE_TARGET is 29M as shown above, and if the maximum PGA allocated is determined to be 235M, then MEMORY_TARGET should be at least 591M (356M + 235M).

3. Choose the value for MEMORY_TARGET that you want to use.

This can be the minimum value that you computed in step 2, or you can choose to use a larger value if you have enough physical memory available.

4. For the MEMORY_MAX_TARGET initialization parameter, decide on a maximum amount of memory that you would want to allocate to the database for the foreseeable future. That is, determine the maximum value for the sum of the System Global Area (SGA) and instance PGA sizes. This number can be larger than or the same as the MEMORY_TARGET value that you chose in the previous step.

5. Do one of the following:

- If you started your Oracle Database instance with a server parameter file, which is the default if you created the database with the Database Configuration Assistant (DBCA), enter the following SQL*Plus command:

  ```
  ALTER SYSTEM SET MEMORY_MAX_TARGET = nM SCOPE = SPFILE;
  ```

  where *n* is the value that you computed in step 4.

  The SCOPE = SPFILE clause sets the value only in the server parameter file, and not for the running instance. You must include this SCOPE clause because MEMORY_MAX_TARGET is not a dynamic initialization parameter.

  Or, you can also select the MEMORY_MAX_TARGET initialization parameter on the Initialization Parameters page in EM Express, click Set, specify a Scope of SPFile, and set a new value, as described in "Viewing and Modifying Initialization Parameters."

- If you started your instance with a text initialization parameter file, manually edit the file so that it contains the following statements:

  ```
  memory_max_target = nM (650M for this example)
  memory_target = mM (591M for this example)
  ```

  where *n* is the value that you determined in step 4, and *m* is the value that you determined in step 3.

Note:

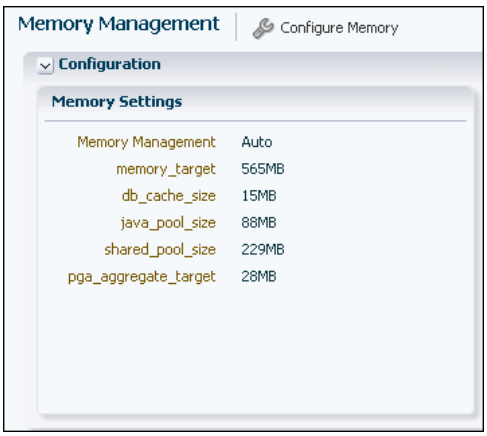
In a text initialization parameter file, if you omit the line for `MEMORY_MAX_TARGET` and include a value for `MEMORY_TARGET`, the database automatically sets `MEMORY_MAX_TARGET` to the value of `MEMORY_TARGET`. If you omit the line for `MEMORY_TARGET` and include a value for `MEMORY_MAX_TARGET`, the `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`.

6. Shut down and restart the database.

   See "Shutting Down and Starting Up the Oracle Instance".

7. In EM Express, from the **Configuration** menu, select **Memory**.

   The Memory Management page appears. In the Memory Settings section, the **Memory Management** value is **Auto**. This indicates that Automatic Memory Management is enabled for the database. The initialization parameter values shown on this page are the ones that have been specified in addition to `MEMORY_MAX_TARGET`.



8. If you started your Oracle Database instance with a server parameter file, make these changes to the following initialization parameter values:

```
MEMORY_TARGET = nM; (591M for this example)
SGA_TARGET = 0;
PGA_AGGREGATE_TARGET = 0;
```

   You can also set these initialization parameter values using the Initialization Parameters page in EM Express, specifying a scope of SPFile. See "Viewing and Modifying Initialization Parameters"for more information.

Note:

The preceding steps instruct you to set SGA_TARGET and PGA_AGGREGATE_TARGET to zero so that the sizes of the SGA and instance PGA are tuned up and down as required, without restrictions. You can omit the statements that set these parameter values to zero and leave either or both of the values as positive numbers. In this case, the values act as minimum values for the sizes of the SGA or instance PGA.

See Also:

"About Memory Management"

Modifying Memory Settings – Automatic Memory Management

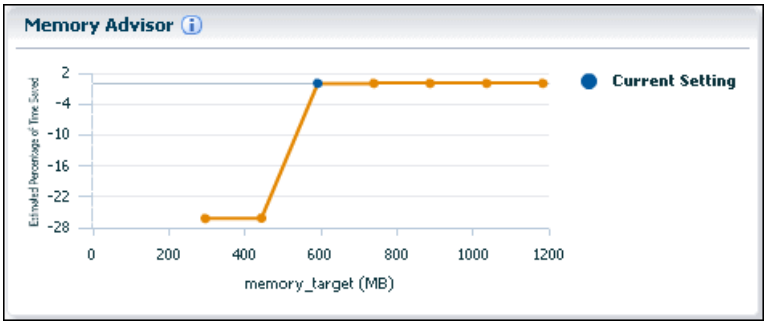
Before you modify memory settings for automatic memory management, use the Memory Advisor graph in Oracle Enterprise Manager Database Express (EM Express) to predict the percentage of time saved by using a different target memory size setting. The steps involved require that you must first have enabled Automatic Memory Management. See "Enabling Automatic Memory Management".

To predict the percentage of time saved for different target memory sizes:

1. In EM Express, from the **Configuration** menu, select **Memory**.

   The Memory Management page appears. In the Memory Settings section, the **Memory Management** value is **Auto**. This indicates that Automatic Memory Management is enabled for the database.

2. Use the Memory Advisor graph (which appears to the right of the Memory Settings section) to predict the percentage of time saved for potential target memory sizes.



   In the Memory Advisor graph:

   - Potential values for the MEMORY_TARGET initialization parameter (in MB) are represented on the horizontal axis of the graph. The current setting of the MEMORY_TARGET initialization parameter is indicated by a blue dot.
   - The corresponding values of time saved are represented on the vertical axis of the graph. The plotted values are expressed as a percentage relative to the current setting of the MEMORY_TARGET initialization parameter.

Negative values represent the percentage of an increase in time consumed (when the memory allotted to Oracle is smaller than the current setting), while positive values represent the percentage of decrease in time consumed (when the memory alloted to Oracle is larger than the current setting.

An orange line on the graph plots different values that can be specified for the MEMORY_TARGET initialization parameter. Click any dot on the orange line to see a prediction of the decrease in time consumed for the MEMORY_TARGET value represented by that dot.

In this figure, the Memory Advisor graph indicates that increasing the current value of the MEMORY_TARGET initialization parameter will not decrease the percentage of time saved.

3. To change the value of the MEMORY_TARGET initialization parameter:

   a. Click **Configure Memory** on the Memory Management page.

      The Initialization Parameter page appears.

   b. Select the MEMORY_TARGET initialization parameter and click **Set**.

      The Set Initialization Parameter page appears.

   c. In the **Scope** field, enter the scope for this change.

      See "Viewing and Modifying Initialization Parameters" for more information about specifying a scope of Memory, or SPFile, or both.

   d. In the **Value** field, enter the new value for the MEMORY_TARGET initialization parameter.

   e. Click **OK**.

      A confirmation message appears.

See Also:

- *Oracle Database Performance Tuning Guide* for more information about memory management methods
- "Optimizing Memory Usage with the Memory Advisors"
- "About Memory Management"

Enabling Automatic Shared Memory Management

This section describes how to change to automatic shared memory management if either automatic memory management or manual shared memory management is currently enabled for your database instance.

To change to automatic shared memory management if automatic memory management is currently enabled:

If automatic memory management is currently enabled, but you would like to have more direct control over the sizes of the System Global Area (SGA) and instance Program Global Area (PGA), you can disable automatic memory management and enable automatic shared memory management. Follow these steps:

1. In Oracle Enterprise Manager Database Express (EM Express), from the **Configuration** menu, select **Initialization Parameters**.

   The Initialization Parameters page appears, with the Current tab displayed.

2. In the Search field, enter `MEMORY_TARGET`.

3. Select `MEMORY_TARGET`, and then click **Set**.

   The Set Initialization Parameter page appears.

4. In the **Value** field, enter 0, specify a **Scope** of **Memory**, and then click **OK**.

   A confirmation message appears.

Note:

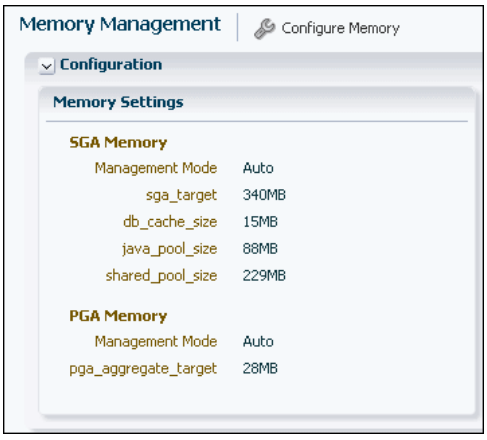
This step changes automatic memory management to automatic shared memory management for the current session. To change automatic memory management to automatic shared memory management and keep it in effect after the database is restarted:

- If your database uses a server parameter file, specify a Scope of SPFile as well as a Scope of Memory on the Set Initialization Parameter page.
- If your database uses a text initialization parameter file, manually set the value of `MEMORY_TARGET` to 0 in that file.

---

5. From the **Configuration** menu, select **Memory**.

   Note that in the SGA Memory subsection of the Memory Settings section, the **Management Mode** value is now **Auto**. This indicates that automatic shared memory management is enabled. The initialization parameter values shown on this page are the ones that have been specified in addition to `MEMORY_TARGET`.



To change to automatic shared memory management if manual shared memory management is currently enabled:

If manual shared memory management is currently enabled, but you would like Oracle Database to help you determine optimal sizes of the SGA and instance PGA, you can disable manual shared memory management and enable automatic shared memory management. Follow these steps:

1. In SQL*Plus, run the following query in the database to obtain a value for SGA_TARGET:

```
SELECT (
   (SELECT SUM(value) FROM V$SGA) -
   (SELECT CURRENT_SIZE FROM V$SGA_DYNAMIC_FREE_MEMORY)
   ) "SGA_TARGET"
FROM DUAL;

SGA_TARGET
----------
 371654656
```

   This value is approximately 354M.

2. In EM Express, from the **Configuration** menu, select **Initialization Parameters**.

   The Initialization Parameters page appears.

3. In the Search field, enter SGA_TARGET.

4. Select SGA_TARGET, and then click **Set**.

   The Set Initialization Parameter page appears.

5. In the **Value** field, enter the SGA_TARGET value from step 1 above (354M in this example), specify a Scope of Memory, and then click **OK**.

   A confirmation message appears.

   Note:

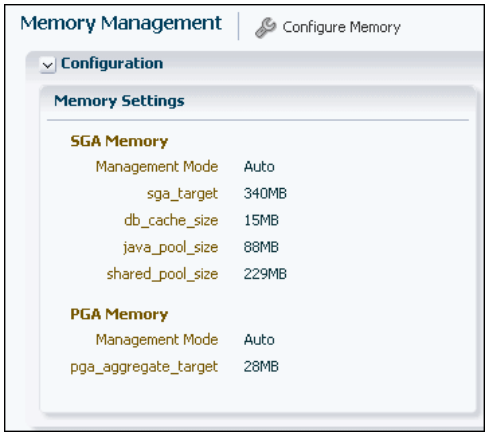
   This step changes manual shared memory management to automatic shared memory management for the current session. To change manual shared memory management to automatic shared memory management and keep it in effect after the database is restarted:

   - If your database uses a server parameter file, specify a Scope of SPFile as well as a Scope of Memory on the Set Initialization Parameter page.
   - If your database uses a text initialization parameter file, manually set the value of MEMORY_TARGET to 0 in that file.

   ---

6. From the **Configuration** menu, select **Memory**.

   Note that in the SGA Memory subsection of the Memory Settings section, the **Management Mode** value is now **Auto**. This indicates that automatic shared memory management is enabled.



7. Do one of the following:

   - For more complete automatic tuning, set the values of the automatically sized SGA components listed in the following table to zero on the Initialization Parameters page:

SGA Component	Initialization Parameter
The shared pool	`SHARED_POOL_SIZE`
The large pool	`LARGE_POOL_SIZE`
The Java pool	`JAVA_POOL_SIZE`
The buffer cache	`DB_CACHE_SIZE`
The Streams pool	`STREAMS_POOL_SIZE`

   - To control the minimum size of one or more automatically sized SGA components, set those component sizes to the desired value, as described in *Oracle Database Administrator's Guide*. Set the values of the other automatically sized SGA components to zero on the Initialization Parameters page.

Modifying Memory Settings – Automatic Shared Memory Management

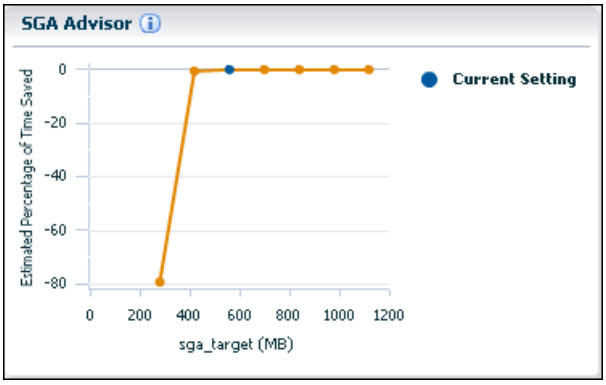
Before you modify memory settings for automatic shared memory management, use the SGA Advisor graph in Oracle Enterprise Manager Database Express (EM Express) to predict the percentage of time saved by using a different total System Global Area (SGA) size. This section assumes that automatic memory management is disabled, and that automatic shared memory management is enabled.

To predict the percentage of time saved for different SGA memory sizes:

1. In EM Express, from the **Configuration** menu, select **Memory**.

   The Memory Management page appears. In the Memory Settings section, the **Management Mode** value is **Auto** in the SGA Memory subsection.This indicates that Automatic Shared Memory Management is enabled for the database.

2. Use the SGA Advisor graph (which appears to the right of the Memory Settings section) to predict the percentage of time saved for potential SGA memory sizes.



In the SGA Advisor graph:

- Potential values for the SGA_TARGET initialization parameter (in MB) are represented on the horizontal axis of the graph. The current setting of the SGA_TARGET initialization parameter is indicated by a blue dot.
- The corresponding values of time saved are represented on the vertical axis of the graph. The plotted values are expressed as a percentage relative to the current setting of the SGA_TARGET initialization parameter.

  Negative values represent the percentage of an increase in time consumed (when the memory allotted to Oracle is smaller than the current setting), while positive values represent the percentage of decrease in time consumed (when the memory alloted to Oracle is larger than the current setting.

An orange line on the graph plots different values that can be specified for the SGA_TARGET initialization parameter. Click any dot on the orange line to see a prediction of the decrease in time consumed for the SGA_TARGET value represented by that dot.

In this figure, the SGA Advisor graph indicates that increasing the current value of the SGA_TARGET initialization parameter will not decrease the percentage of time saved.

3. To change the value of the SGA_TARGET initialization parameter:

   a. Click **Configure Memory** on the Memory Management page.

      The Initialization Parameter page appears.

   b. Select the SGA_TARGET initialization parameter and click **Set**.

      The Set Initialization Parameter page appears.

   c. In the **Scope** field, enter the scope for this change.

      See "Viewing and Modifying Initialization Parameters" for more information about setting a scope of Memory, or SPFile, or both.

   d. In the **Value** field, enter the new value for the SGA_TARGET initialization parameter.

   e. Click **OK**.

      A confirmation message appears.

See Also:

- *Oracle Database Performance Tuning Guide* for more information about memory management modes
- "Optimizing Memory Usage with the Memory Advisors"
- "About Memory Management"

Enabling Manual Shared Memory Management

Follow these steps to enable manual shared memory management:

1. In Oracle Enterprise Manager Database Express (EM Express), from the **Configuration** menu, select **Initialization Parameters**.

   The Initialization Parameters page appears.

2. In the Search field, enter `SGA_TARGET`.
3. Select `SGA_TARGET`, and then click **Set**.

   The Set Initialization Parameter page appears.

4. In the **Value** field, enter 0, specify a **Scope** of **Memory**, and then click **OK**.

   A confirmation message appears.

   Note:

   This step sets `SGA_TARGET` to 0 for the current session. To set `SGA_TARGET` to 0 and keep it in effect after the database is restarted:

   - If your database uses a server parameter file, specify a Scope of SPFile as well as a Scope of Memory on the Set Initialization Parameter page.
   - If your database uses a text initialization parameter file, manually set the value of `SGA_TARGET` to 0 in that file.

5. In the Search field, enter `MEMORY_TARGET`.
6. Select `MEMORY_TARGET`, and then click **Set**.

   The Set Initialization Parameter page appears.

7. In the **Value** field, enter 0, specify a Scope of Memory, and then click **OK**.

   A confirmation message appears.

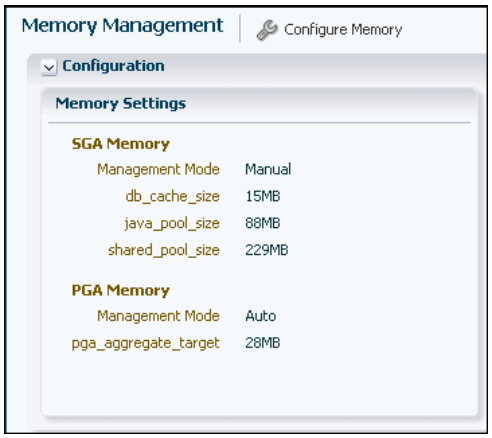
Note:

This step sets `MEMORY_TARGET` to 0 for the current session. To set `MEMORY_TARGET` to 0 and keep it in effect after the database is restarted:

- If your database uses a server parameter file, specify a Scope of SPFile as well as a Scope of Memory on the Set Initialization Parameter page.
- If your database uses a text initialization parameter file, manually set the value of `MEMORY_TARGET` to 0 in that file.

8. From the **Configuration** menu, select **Memory**.

   Note that under the SGA Memory section, the **Management Mode** value is now **Manual**. This indicates that manual shared memory management is enabled.



9. For details on setting values for the various SGA components, see *Oracle Database Administrator's Guide*.

Modifying Memory Settings - Manual Shared Memory Management

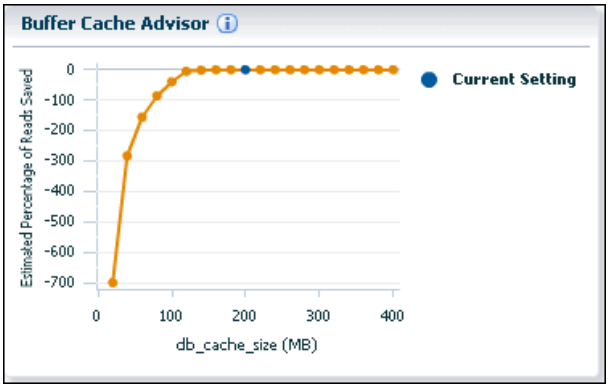
Before you modify memory settings for manual shared memory management, use the Buffer Cache Advisor graph in Oracle Enterprise Manager Database Express (EM Express) to predict the estimated percentage of reads saved by using a different database cache size. This section assumes that automatic memory management is disabled, and that manual shared memory management is enabled.

To predict the percentage of reads saved for different database cache sizes:

1. In EM Express, from the **Configuration** menu, select **Memory**.

   The Memory Management page appears. In the Memory Settings section, the **Management Mode** value is **Manual** in the SGA Memory subsection.This indicates that Manual Shared Memory Management is enabled for the database.

2. Use the Buffer Cache Advisor graph (which appears to the right of the Memory Settings section) to predict the percentage of reads saved for potential database cache sizes.



In the Buffer Cache Advisor graph:

- Potential values for the DB_CACHE_SIZE initialization parameter (in MB) are represented on the horizontal axis of the graph. The current setting of the DB_CACHE_SIZE initialization parameter is indicated by a blue dot.
- The corresponding values of reads saved are represented on the vertical axis of the graph. The plotted values are expressed as a percentage relative to the current setting of the DB_CACHE_SIZE initialization parameter.

  Negative values represent the percentage of an increase in reads (when the memory allotted to Oracle is smaller than the current setting), while positive values represent the percentage of decrease in reads (when the memory alloted to Oracle is larger than the current setting.

An orange line on the graph plots different values that can be specified for the DB_CACHE_SIZE initialization parameter. Click any dot on the orange line to see a prediction of the percentage of reads saved for the DB_CACHE_SIZE value represented by that dot.

In this figure, the Buffer Cache Advisor graph indicates that increasing the current value of the DB_CACHE_SIZE initialization parameter will not increase the percentage of reads saved.

3. To change the value of the DB_CACHE_SIZE initialization parameter:

   a. Click **Configure Memory** on the Memory Management page.

      The Initialization Parameter page appears.

   b. Select the DB_CACHE_SIZE initialization parameter and click **Set**.

      The Set Initialization Parameter page appears.

   c. In the **Scope** field, enter the scope for this change.

      See "Viewing and Modifying Initialization Parameters" for more information about specifying a scope of MEMORY, or SPFile, or both.

   d. In the **Value** field, enter the new value for the DB_CACHE_SIZE initialization parameter.

   e. Click **OK**.

      A confirmation message appears.

Modifying Memory Settings – Automatic PGA Memory Management

Modifying memory settings for automatic Program Global Area (PGA) memory management involves using the PGA Advisor graph in Oracle Enterprise Manager Database Express (EM Express) to modify the instance PGA size. This section assumes that automatic memory management is disabled, and that automatic PGA memory management is enabled.

Note:

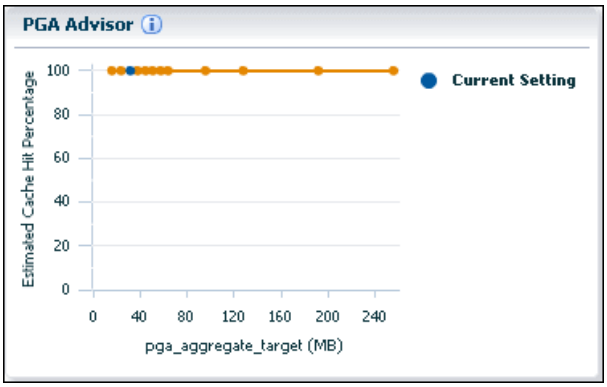
When you disable automatic memory management, automatic PGA memory management remains enabled by default.

To modify the instance PGA size:

1. In EM Express, from the **Configuration** menu, select **Memory**.

   The Memory Management page appears. If the database has either automatic shared memory management or manual shared memory management enabled, then the PGA Advisor Graph appears as the second chart to the right of the Memory Settings section.

2. Use the PGA Advisor graph to predict the estimated cache hit percentage for potential database cache sizes.



   In the PGA Advisor graph:

   - Potential values for the `PGA_AGGREGATE_TARGET` initialization parameter are represented on the horizontal axis of the graph. The unit size (for example, `MB` or `GB`) is also indicated on the horizontal axis. The current setting of the `PGA_AGGREGATE_TARGET` initialization parameter is indicated by a blue dot.
   - The corresponding estimated cache hit percentage values are represented on the vertical axis of the graph. The plotted values are expressed as a percentage relative to the current setting of the `PGA_AGGREGATE_TARGET` initialization parameter.

   An orange line on the graph plots different values that can be specified for the `PGA_AGGREGATE_TARGET` initialization parameter. Click any dot on the orange line to see an estimate of the percentage of cache hits for the `PGA_AGGREGATE_TARGET` value represented by that dot.

In this figure, the PGA Advisor graph indicates that increasing the current value of the PGA_AGGREGATE_TARGET initialization parameter will not increase the percentage of cache hit.

3. To change the value of the PGA_AGGREGATE_TARGET initialization parameter:

   a. Click **Configure Memory** on the Memory Management page.

      The Initialization Parameters page appears.

   b. Select the PGA_AGGREGATE_TARGET initialization parameter and click **Set**.

      The Set Initialization Parameter page appears.

   c. In the **Scope** field, enter the scope for this change.

      See "Viewing and Modifying Initialization Parameters" for more information about specifying a scope of Memory, or SPFile, or both.

   d. In the **Value** field, enter the new value for the PGA_AGGREGATE_TARGET initialization parameter.

   e. Click **OK**.

      A confirmation message appears.

See Also:

- *Oracle Database Performance Tuning Guide* for more information about memory management modes
- "About Memory Management"

Instances: Oracle By Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter, and includes annotated screenshots.

To view the Managing the Oracle Instance OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6285,1
```

6

Managing Database Storage Structures

This chapter discusses using Oracle Enterprise Manager Database Express (EM Express) to view and manage the storage structures of your database. This chapter contains the following sections:

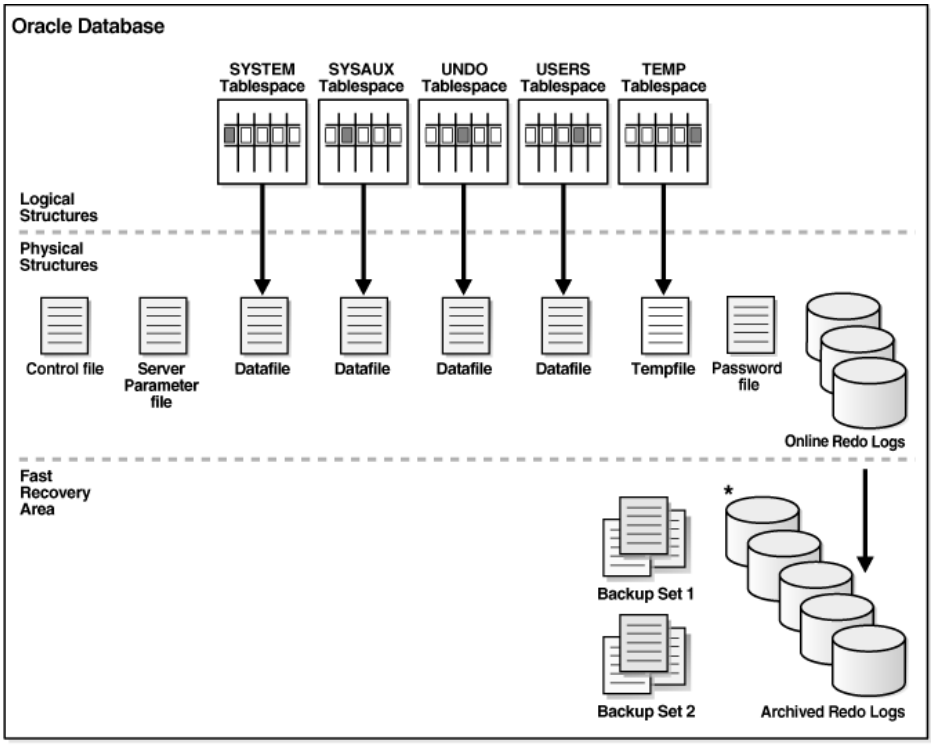
About Database Storage Structures

An Oracle database is made up of physical and logical structures. Physical structures can be seen and operated on from the operating system, such as the physical files that store data on a disk.

Logical structures are created and recognized by Oracle Database and are not known to the operating system. The primary logical structure in a database, a tablespace, contains physical files. The applications developer or user may be aware of the logical structure, but is not usually aware of this physical structure. The database administrator (DBA) must understand the relationship between the physical and logical structures of a database.

Figure 1 shows the relationships between logical and physical structures. This figure also shows recovery-related structures that are optionally kept in the fast recovery area. See "Fast Recovery Area" for more information.

Figure 1 Oracle Database Storage Structures



Oracle Database can automate much of the management of its structure. Oracle Enterprise Manager Database Express (EM Express) provides a Web-based graphical user interface (GUI) to enable easier management and monitoring of your database.

From a physical perspective, a multitenant container database (CDB) has basically the same structure as a non-CDB, except that each pluggable database (PDB) has its own set of tablespaces (including its own `SYSTEM` and `SYSAUX` tablespaces) and data files.

A CDB contains the following files:

- One control file
- One online redo log
- One or more sets of temp files
- One set of undo data files
- A set of system data files for every container
- Zero or more sets of user-created data files

This section provides background information about the various database storage structures. It contains the following topics:

- About Control Files
- About Online Redo Log Files
- About Archived Redo Log Files
- About Rollback Segments

- About Data Files
- About Tablespaces
- About Other Storage Structures

See Also:

- "Viewing Database Storage Structure Information"
- *Oracle Database Concepts* for more information about database storage structures
- *Oracle Database Concepts* for an introduction to CDBs and PDBs
- *Oracle Database Concepts* for more information about database files in a CDB

About Control Files

A control file tracks the physical components of the database. It is the root file that the database uses to find all the other files used by the database. Because of the importance of the control file, Oracle recommends that the control file be **multiplexed**, or have multiple identical copies. For databases created with Oracle Database Configuration Assistant (DBCA), two copies of the control file are automatically created and kept synchronized with each other.

If any control file fails, then your database becomes unavailable. If you have a control file copy, however, you can shut down your database and re-create the failed control file from the copy, then restart your database. Another option is to delete the failed control file from the `CONTROL_FILES` initialization parameter and restart your database using the remaining control files.

See Also:

- *Oracle Database Concepts* for an overview of control files
- *Oracle Database Administrator's Guide* for detailed information about control files
- "Viewing Control File Information"

About Online Redo Log Files

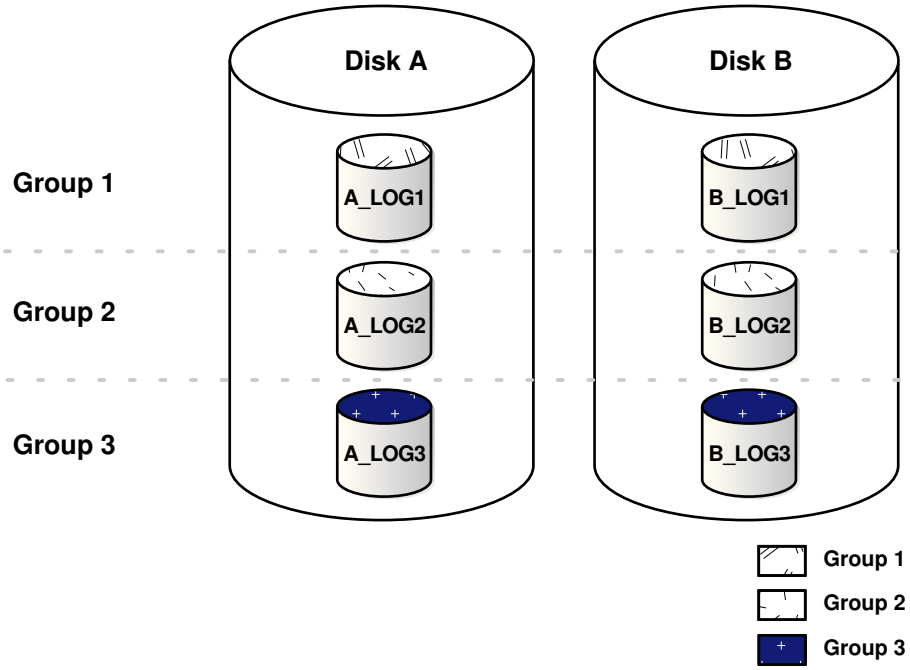
Every Oracle database has a set of two or more online redo log files. The set of online redo log files is collectively known as the **redo log** for the database. A redo log is made up of redo entries, which are also called **redo records**.

The online redo log stores a copy of the changes made to data. If a failure requires a data file to be restored from backup, then the recent data changes that are missing from the restored data file can be obtained from the online redo log files, so work is never lost. The online redo log files are used to recover a database after hardware, software, or media failure. To protect against a failure involving the online redo log file itself, Oracle Database can multiplex the online redo log file so that two or more identical copies of the online redo log file can be maintained on different disks.

The online redo log for a database consists of groups of online redo log files. A group consists of an online redo log file and its multiplexed copies. Each identical copy is considered to be a member of that group. Each group is defined by a number, such as Group 1.

Figure 2 shows the configuration of a database that has three online redo log groups and two members in each group. For each group, the members are stored on separate disks for maximum availability. For example, the members of Group 1 are the redo log files `A_LOG1` and `B_LOG1`.

Figure 2 Online Redo Log Groups and Their Members



The database log writer process (LGWR) writes redo records from the memory buffer to a redo log group until the log files in that group reach their storage size limit, or until you request a log switch operation. The LGWR process then writes to the next log group. The LGWR process performs this action in a circular fashion so that the oldest group is overwritten by the most recent redo records.

See Also:

- *Oracle Database Concepts* for an overview of online redo logs
- *Oracle Database Administrator's Guide* for detailed information about redo logs
- "Viewing Online Redo Log File Information"
- "Switching a Log File"
- "Managing the Online Redo Log"

About Archived Redo Log Files

When you enable archiving of the online redo logs, Oracle Database copies the online redo log files to another location before they are overwritten. These copied files are referred to as **archived redo log files**. You can archive to multiple locations.

These archived redo log files extend the amount of redo data that can be saved and are used for recovery. Archived redo log files are required to recover a backup of the database from the time of the backup to the current time. Archiving can be either enabled or disabled for the database, but Oracle strongly recommends that you enable archiving. Oracle also recommends that you configure the database to write archived redo log files to the fast recovery area.

See Also:

- *Oracle Database Concepts* for an overview of archived redo log files
- *Oracle Database Administrator's Guide* for detailed information about archived redo log files
- "Viewing Archived Redo Log File Information"
- "Configuring Recovery Settings" for information about enabling redo log archiving
- "Fast Recovery Area" for background information about the fast recovery area

About Rollback Segments

Rollback segments were database structures used to track undo information for the database in earlier releases of Oracle Database. Now, the preferred way of managing undo information is with the undo tablespace. For more information, see "Managing Undo Data".

Note:

Oracle Database uses a `SYSTEM` rollback segment for performing system transactions. It is created automatically when the database is created, and is always brought online at instance startup. It is located in the `SYSTEM` tablespace. You are not required to perform any operations to manage the `SYSTEM` rollback segment.

About Data Files

Data files are the operating system files that store the data within the database. The data is written to these files in an Oracle proprietary format that cannot be read by other programs. **Tempfiles** are a special class of data files that are associated only with temporary tablespaces.

Data files can be broken down into the following components:

- Segment

A **segment** contains a specific type of database object. For example, a table is stored in a table segment, and an index is stored in an index segment. A data file can contain many segments.

- Extent

  An **extent** is a contiguous set of data blocks within a segment. Oracle Database allocates space for segments in units of one extent. When the existing extents of a segment are full, the database allocates another extent for that segment.

- Data block

  A **data block**, also called a **database block**, is the smallest unit of I/O to database storage. An extent consists of several contiguous data blocks. The database uses a default block size at database creation.

  After the database has been created, it is not possible to change the default block size without re-creating the database. It is possible, however, to create a tablespace with a block size different than the default block size.

Segments, extents, and data blocks are all logical structures. Only Oracle Database can determine how many data blocks are in a file. The operating system recognizes only files and operating system blocks, not the number of data blocks in an Oracle Database file. Each data block maps to one or more operating system blocks.

See Also:

- *Oracle Database Administrator's Guide* for detailed information about data files
- *Oracle Database Concepts* for more information about segments, extents, and blocks

About Tablespaces

A database is divided into logical storage units called **tablespaces**, which group related logical structures (such as tables, views, and other database objects). For example, all application objects can be grouped into a single tablespace to simplify maintenance operations.

A tablespace consists of one or more physical data files. Database objects assigned to a tablespace are stored in the physical data files of that tablespace.

When you create an Oracle database, some tablespaces already exist, such as `SYSTEM` and `SYSAUX`.

Tablespaces provide a means to physically locate data on storage. When you define the data files that comprise a tablespace, you specify a storage location for these files. For example, you might specify a data file location for a certain tablespace as a designated host directory (implying a certain disk volume) or designated Oracle Automatic Storage Management disk group. Any schema objects assigned to that tablespace then get located in the specified storage location. Tablespaces also provide a unit of backup and recovery. The backup and recovery features of Oracle Database enable you to back up or recover at the tablespace level.

Table 1 describes some tablespaces included in the database.

Table 1 Tablespaces and Descriptions

Tablespace	Description
EXAMPLE	This tablespace contains the sample schemas that are included with Oracle Database. The sample schemas provide a common platform for examples. Oracle documentation and educational materials contain examples based on the sample schemas.
SYSTEM	This tablespace is automatically created at database creation. Oracle Database uses it to manage the database. It contains the data dictionary, which is the central set of tables and views used as a read-only reference for a particular database. It also contains various tables and views that contain administrative information about the database. These are all contained in the SYS schema, and can be accessed only by the SYS user or other administrative users with the required privilege.
SYSAUX	This is an auxiliary tablespace to the SYSTEM tablespace. The SYSAUX tablespace contains data for some components and products, reducing the load on the SYSTEM tablespace. Every database using Oracle Database 10*g* release 1 (10.1) or later must have a SYSAUX tablespace. Components that use SYSAUX as their default tablespace during installation include Automatic Workload Repository, Oracle Streams, and Oracle Text. For more information, see *Oracle Database Administrator's Guide*.
TEMP	This tablespace stores temporary data generated when processing SQL statements. For example, this tablespace would be used for query sorting. Every database should have a temporary tablespace that is assigned to users as their temporary tablespace. In the preconfigured database, the TEMP tablespace is specified as the default temporary tablespace. If no temporary tablespace is specified when a user account is created, then Oracle Database assigns this tablespace to the user.
UNDOTBS1	This is the undo tablespace used by the database to store undo information. See "Managing Undo Data" to understand how an Oracle database uses the undo tablespace. Every database must have an undo tablespace.
USERS	This tablespace is used to store permanent user objects and data. Similar to the TEMP tablespace, every database should have a tablespace for permanent user data that is assigned to users. Otherwise, user objects will be created in the SYSTEM tablespace, which is not good practice. In the preconfigured database, USERS is designated as the default tablespace for all new users.

You can create new tablespaces to support your user and application data requirements. During tablespace creation, you set the following parameters:

- Locally Managed Tablespaces
- Tablespace Types
- Tablespace Status
- Autoextend Tablespace
- Encrypted Tablespaces

See Also:

- "Viewing Tablespace and Data File Information"
- "Creating a Tablespace"
- "Modifying a Tablespace"
- *Oracle Database Concepts* for an overview of tablespaces

Locally Managed Tablespaces

Space management within a tablespace involves keeping track of available (free) and used space, so that space is allocated efficiently during data insertion and deletion. Locally managed tablespaces keep the space allocation information within the tablespace, not in the data dictionary, thus offering better performance. By default, Oracle Database sets all newly created tablespaces to be locally managed with automatic segment management, a feature that further improves performance.

Tablespace Types

There are three types of tablespaces:

- Permanent

  You use permanent tablespaces to store your user and application data. Oracle Database uses permanent tablespaces to store permanent data, such as system data. Each user is assigned a default permanent tablespace.

- Undo

  A database running in automatic undo management mode transparently creates and manages undo data in the undo tablespace. Oracle Database uses undo data to roll back transactions, to provide read consistency, to help with database recovery, and to enable features such as Oracle Flashback Query. A database instance can have only one active undo tablespace.

- Temporary

  Temporary tablespaces are used for storing temporary data, as would be created when SQL statements perform sort operations. An Oracle database gets a temporary tablespace when the database is created. You would create another temporary tablespace if you were creating a temporary tablespace group. Under typical circumstances, you do not have to create additional temporary tablespaces. If you have an extremely large database, then you might configure additional temporary tablespaces.

  The physical files that comprise a temporary tablespace are called tempfiles, as opposed to data files.

  The `TEMP` tablespace is typically used as the default temporary tablespace for users who are not explicitly assigned a temporary tablespace.

See Also:

- *Oracle Database Administrator's Guide* to learn more about temporary tablespaces
- *Oracle Database Concepts* for more information about undo tablespaces

Tablespace Status

You can set tablespace status as follows:

- Read Write

  Users can read and write to the tablespace after it is created. This is the default.

- Read Only

  If the tablespace is created Read Only, then the tablespace cannot be written to until its status is changed to Read Write. It is unlikely that you would create a Read Only tablespace, but you might change it to that status after you have written data to it that you do not want modified.

- Offline

  If the tablespace has a status of Offline, then no users can access it. You might change the status of a tablespace to Offline before performing maintenance or recovery on the data files associated with that tablespace.

Autoextend Tablespace

You can set a tablespace to automatically extend itself by a specified amount when it reaches its size limit. If you do not enable autoextend, then you are alerted when the tablespace reaches its critical or warning threshold size. The critical and warning threshold parameters have default values that you can change at any time. These parameters also cause alerts to be generated for autoextending tablespaces that are approaching their specified size limit. You can respond to size alerts by manually increasing the tablespace size. You do so by increasing the size of one or more of the tablespace data files or by adding another data file to the tablespace.

Note:

Although it is common to refer to tablespaces as autoextending, automatic extension is a data file property, not a tablespace property. That is, when you create the data files that comprise a tablespace, you indicate whether these data files automatically extend. A tablespace that has autoextending data files is considered to be an autoextending tablespace. You can specify a maximum size for an autoextending data file.

Encrypted Tablespaces

Encrypted tablespaces primarily protect your data from unauthorized access by means other than through the database. For example, when encrypted tablespaces are written to backup media for travel from one Oracle database to another or for travel to an off-site facility for storage, they remain encrypted. Also, encrypted tablespaces protect data from users who try to circumvent the security features of the database and access database files directly through the operating system file system.

You can encrypt any permanent tablespace to protect sensitive data. When you encrypt a tablespace, all tablespace blocks are encrypted. All segment types are supported for encryption, including tables, clusters, indexes, LOBs, table and index partitions, and so on. Tablespace encryption is completely transparent to your applications, so no application modification is necessary.

See Also:

Oracle Database Security Guide for more information about tablespace encryption

About Other Storage Structures

Other storage structures that can exist in an Oracle database include the initialization parameter file, the password file, and backup files.

Initialization Parameter File

Initialization parameters are used by the Oracle instance at startup to determine the run-time properties and resources for the database. Some parameters can be set or modified while the database is running. Other initialization parameters require the database to be restarted for the changes to take effect. See "Viewing and Modifying Initialization Parameters."

Password File

A database can use a password file to authenticate administrative users with SYSDBA, SYSOPER, and SYSBACKUP privileges. A password file is required for remote connections to the database with any of these privileges. These privileges enable a DBA to start and shut down the database, back up and recover the database, and perform other high-level administrative tasks. This password file is outside of the database itself, thereby enabling the authentication of a DBA when the database is not yet started. (A DBA must authenticate before starting the database.)

When you invoke DBCA as part of the Oracle Database installation process, DBCA creates a password file with one entry: the SYS user. Granting SYSDBA, SYSOPER, or SYSBACKUP to a user adds that user to the password file automatically.

Note:

Oracle Database can also use operating system authentication to authenticate users with the SYSDBA, SYSOPER, and SYSBACKUP privileges. Operating system authentication takes precedence over password file authentication. See "Starting SQL*Plus and Connecting to the Database."

Backup Files

Backup files are not technically database files, but are copies of the database in some form that can be used to recover the database if a failure causes loss of data.

See Also:

- Managing the Oracle Instance for more information about initialization parameters and the initialization parameter file
- *Oracle Database Concepts* for more information about password files and operating system authentication
- "SYSDBA and SYSOPER System Privileges"
- Performing Backup and Recovery for more information about backup files
- *Oracle Database Concepts* for an overview of backup and recovery

Viewing Database Storage Structure Information

To assist you in managing the storage structures within your database, this section provides instructions for viewing information about the various database storage structures using Oracle Enterprise Manager Database Express (EM Express).

This section contains the following topics:

- Viewing Control File Information
- Viewing Online Redo Log File Information
- Viewing Archived Redo Log File Information
- Viewing Tablespace and Data File Information

See Also:

"About Database Storage Structures"

Viewing Control File Information

You can use Oracle Enterprise Manager Database Express (EM Express) to view location and status information about control files.

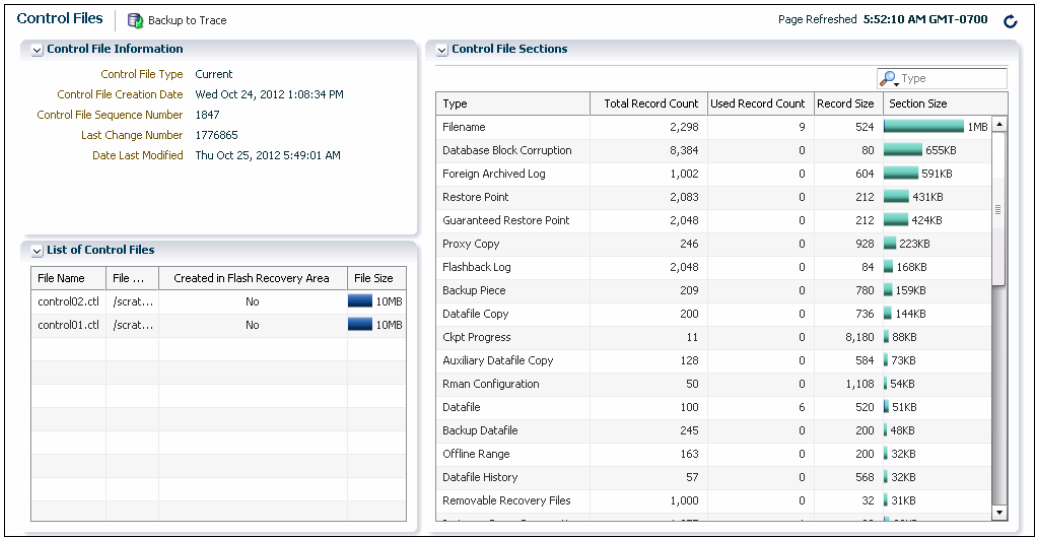
Note:

In a multitenant container database (CDB), EM Express provides control file information only in the root, not in the pluggable databases (PDBs).

To view control file information:

1. In EM Express, from the **Storage** menu, select **Control Files**.

   The Control Files page appears.



This page shows whether your database has a multiplexed control file. The List of Control Files and Control Files sections give you more detailed information about your control files.

2. To back up the current control file to a trace file, click **Backup to Trace**.

   See *Oracle Database Backup and Recovery User's Guide* for more information about backing up the control file to a trace file.

See Also:

"About Control Files"

Viewing Online Redo Log File Information

You can use Oracle Enterprise Manager Database Express (EM Express) to view status and multiplexing information about online redo log files.

Note:

In a multitenant container database (CDB), EM Express provides online redo log file information only in the root, not in the pluggable databases (PDBs).

To view online redo log file information:

1. In EM Express, from the **Storage** menu, select **Redo Log Groups**.

   The Redo Log Groups page appears. This page shows the attributes of the online redo log groups for your database.

   When an online redo log group contains only one member, it is not multiplexed. Note the Status attribute for the online redo log groups. The online redo log group with status Current is the one currently being written to disk by the log writer.

2. (Optional) Expand a redo log group number to view information about online redo log group members.

See Also:

- "About Online Redo Log Files"
- "Managing the Online Redo Log"

Viewing Archived Redo Log File Information

You can use Oracle Enterprise Manager Database Express (EM Express) to view status information about archived redo log files.

Note:

In a multitenant container database (CDB), EM Express provides archived redo log file information only in the root, not in the pluggable databases (PDBs).

Note:

Archived redo log files do not exist until you set the database in `ARCHIVELOG` mode.

To view archived redo log file information:

1. In EM Express, from the **Storage** menu, select **Archive Logs**.

   The Archive Logs page appears.

2. View information on the Archive Logs page about the redo logs that have been archived.

See Also:

- "About Archived Redo Log Files"
- "Configuring Your Database for Basic Backup and Recovery"

Viewing Tablespace and Data File Information

You can use Oracle Enterprise Manager Database Express (EM Express) to view configuration, size, and status information about tablespaces and data files.

Note:

In a multitenant container database (CDB), EM Express provides tablespace and data file information only in the pluggable databases (PDBs), not in the root.

To view tablespace information:

1. In EM Express, from the **Storage** menu, select **Tablespaces**.

   The Tablespaces page appears.

2. To view information about the data files for a particular tablespace, expand that tablespace. Information about the data files appears underneath the tablespace.

See Also:

- "About Tablespaces"
- "Modifying a Tablespace"

Performing Common Database Storage Tasks

As data is added to your database, the tablespace requirements for your database change. As a database administrator (DBA), you can use Oracle Enterprise Manager Database Express (EM Express) to perform the following tasks to effectively manage the tablespaces and database storage:

- Creating a Tablespace
- Modifying a Tablespace
- Dropping a Tablespace

Creating a Tablespace

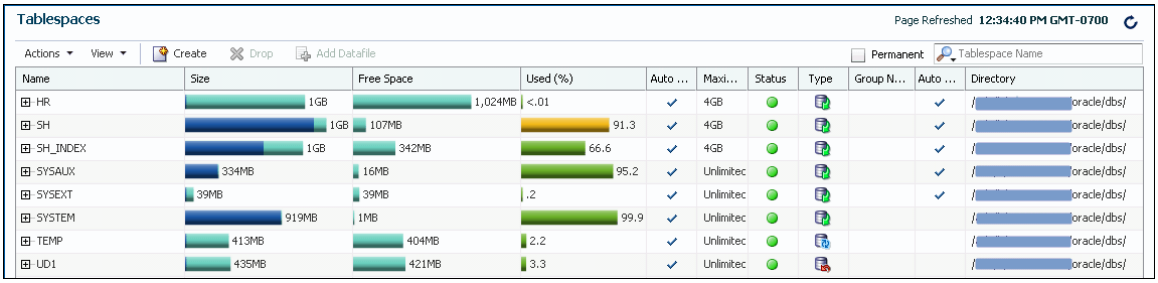
You can create additional tablespaces to store user data, so that not all data is stored in the USERS tablespace. The following are some reasons to create additional tablespaces:

- For certain users, groups of users, or applications, it may be convenient to keep all application data in a separate tablespace or set of tablespaces for backup and recovery or maintenance reasons. For example, suppose you must recover all application data from backup due to a hardware or software failure, and you want to perform an offline recovery. If the application data is kept in a separate tablespace, then you can take just that tablespace offline and recover it, without affecting the operation of other database applications.
- Some applications, such as those with large partitioned tables, may benefit from distributing data across multiple tablespaces. This approach allows the optimal use of the available storage because frequently accessed data can be placed on high performance disks, and infrequently retrieved data can be placed on less expensive storage.

To create a tablespace:

1. In Oracle Enterprise Manager Database Express (EM Express), from the **Storage** menu, select **Tablespaces**.

   The Tablespaces page appears.



2. To create a new tablespace, click the **Create** button.

   The Create Tablespace wizard appears, showing the General page.

3. In the **Name** field, enter a name for the tablespace.

4. In the Bigfile section, select **Smallfile**.

   Note:

   If you select **Use bigfile tablespace**, then the tablespace can have only one data file. Bigfile tablespaces are used with very large databases that use Oracle Automatic Storage Management or other logical volume managers that support striping, RAID, and dynamically extensible logical volumes.

   EM Express does not support Oracle Automatic Storage Management database instances.

   ---

5. In the Status section, select **Online**.

6. To go to the next page in the wizard, click the right arrow button.

   The Add Datafiles page appears.

7. For the **Datafiles** field, enter the name for the datafile. If the datafile name includes a number in the suffix (such as `df_1`), you can click the + button or press the Enter key to create multiple data files with the options you select on the Add Datafiles page.

8. For the **File Size** field, enter appropriate values for your data file location and initial size.

9. Select **Auto Extend**.

10. For the **Increment** field, select the additional space to be added to the file each time it extends.

11. For the **Maximum File Size** field, enter the maximum size for this data file.

12. After adding the data files for the new tablespace, click the right arrow button to go to the next page in the wizard.

    The Space page appears.

Note:

On the Add Datafiles page and subsequent pages in the wizard, you can click the right arrow button to go to the next page, or you can click **OK** to create the tablespace without continuing through the wizard. The default values on on the remaining wizard pages will be used to create the tablespace.

13. For **Block Size**, select the block size to use for the tablespace.
14. For **Extent Allocation**, select **Automatic**.
15. Click the right arrow button to go to the next page in the wizard.

    The Logging page appears.
16. In the Logging section, select **Logging**.
17. Click the right arrow button to go to the final page in the wizard.

    The Segments page appears.
18. In the Segment Space Management section, select **Automatic**.
19. In the Compression section, select **None**.
20. Click **OK** to add the tablespace.

See Also:

"About Tablespaces"

Modifying a Tablespace

You can use Oracle Enterprise Manager Database Express (EM Express) to modify a tablespace. For example, you can extend it by increasing data file sizes or adding another data file, set it to automatically extend, change its space usage alert thresholds, or change its status to Offline.

This section contains the following topics:

- Setting a Tablespace to Automatically Extend
- Setting the Datafile for a Smallfile Tablespace to Automatically Extend
- Taking a Tablespace Offline

See Also:

"About Tablespaces"

Setting a Tablespace to Automatically Extend

You can use Oracle Enteprise Manager Database Express (EM Express) to set a tablespace to automatically extend when it reaches its size limit. The following instructions assume that the tablespace was previously not an autoextending tablespace.

Note:

Only bigfile tablespaces can be automatically extended. However, individual datafiles for a smallfile tablespace can be automatically extended. See "Setting the Datafile for a Smallfile Tablespace to Automatically Extend" for instructions for setting up individual data files for a smallfile tablespace to automatically extend.

To set a tablespace to automatically extend:

1. In EM Express, from the **Storage** menu, select **Tablespaces**.

   The Tablespaces page appears.

2. Select the bigfile tablespace for which you want to enable autoextend, and then click **Edit Auto Extend**.

   The Auto Extend Setting of Bigfile Tablespace page appears.

3. Complete the following steps:

   a. Select **Auto Extend**.

   b. Set a suitable increment, such as 10 MB.

      This is the amount of disk space that is added to the data file when it needs more storage space.

   c. For Maximum File Size, enter a value in KB, MB, GB, or TB, depending on available storage.

4. Click **OK**.

   A confirmation message appears.

Setting the Datafile for a Smallfile Tablespace to Automatically Extend

You can use Oracle Enterprise Manager Database Express (EM Express) to set a datafile for a smallfile tablespace to automatically extend when it reaches its size limit. The following instructions assume that the datafile was previously not an autoextending datafile.

To set a datafile to automatically extend:

1. In EM Express, from the **Storage** menu, select **Tablespaces**.

   The Tablespaces page appears.

2. For a smallfile tablespace, select a datafile for which you want to enable autoextend, and then from the **Actions** menu, select **Edit Auto Extend**.

   The Auto Extend Setting of Datafile page appears.

3. Complete the following steps:

   a. Select **Auto Extend**.

   b. Set a suitable increment, such as 10 MB.

This is the amount of disk space that is added to the data file when it needs more storage space.

c. For Maximum File Size, enter a value in KB, MB, GB, or TB, depending on available storage.

4. Click **OK**.

   A confirmation message appears.

5. You can perform these steps for all the datafiles in a smallfile tablespace to set all of the datafiles to automatically extend.

Taking a Tablespace Offline

You can use Oracle Enterprise Manager Database Express (EM Express) to take a tablespace offline. You may want to take a tablespace offline for any of the following reasons:

- To make a portion of the database unavailable while still allowing access to the remainder of the database
- To make an application and its group of tables temporarily unavailable while updating or maintaining the application
- To perform an offline tablespace backup (even though a tablespace can be backed up while online and in use)
- To recover a tablespace after a hardware or software failure
- To rename or relocate tablespace data files

To take a tablespace offline:

1. In EM Express, from the **Storage** menu, select **Tablespaces**.

   The Tablespaces page appears.

2. Select the tablespace that you want to take offline. From the **Actions** menu, select **Set Status**, then **Take Offline**.

   The Bring Tablespace Offline page appears.

3. For **Offline Options**, select **Normal**.

4. Click **OK**.

   A confirmation message appears.

Note:

To bring the tablespace back online, select the tablespace on the Tablespaces page. Then from the **Actions** menu, select **Set Status**, then **Place Online**.

See Also:

Oracle Database Administrator's Guide for more information about taking tablespaces offline and for information about renaming or relocating data files.

Dropping a Tablespace

You can use Oracle Enterprise Manager Database Express (EM Express) to drop (delete) a tablespace. After a tablespace has been dropped, the objects and data in it are no longer available. To recover them can be a time-consuming process. Oracle recommends performing a backup before and after dropping a tablespace.

To drop a tablespace:

1. In EM Express, from the **Storage** menu, select **Tablespaces**.

   The Tablespaces page appears.

2. Select the tablespace to drop, and then click **Drop**.

   The Drop Tablespace page appears.

   EM Express asks for confirmation to delete the tablespace and gives you the option to also delete the associated data files from the disk.

3. Choose from the following options and then click **OK**:

   - **Drop Contents**
   - **Drop Datafiles**
   - **Drop Constraints**

4. A confirmation is displayed and the deleted tablespace no longer appears on the Tablespaces page.

See Also:

"About Tablespaces"

Managing the Online Redo Log

The online redo log files are a critical component in database recovery. Every transaction in the database updates the redo logs, regardless of whether archiving is enabled. During crash, instance, or media recovery, the database properly applies redo log files in ascending order by using the log sequence number of the necessary archived and redo log files.

If properly configured, the online redo logs require little maintenance. This section describes the more common redo log management tasks. It contains the following topics:

Note:

In a multitenant container database (CDB), Oracle Enterprise Manager Database Express (EM Express) provides online redo log file information only in the root, not in the pluggable databases (PDBs).

- Multiplexing the Online Redo Log
- Switching a Log File

See Also:

- "About Online Redo Log Files"
- "Viewing Online Redo Log File Information"

Multiplexing the Online Redo Log

Oracle recommends that you multiplex the online redo log. Multiplexing provides better protection for data if an instance or media failure occurs. You can multiplex the online redo log using Oracle Enterprise Manager Database Express (EM Express).

To multiplex your online redo log, you must add members to each online redo log group. It is not required that online redo log groups be symmetrical, but Oracle recommends that your groups all have the same number of members. A database must have a minimum of two online redo log groups.

Note:

When you multiplex the online redo log, the database must increase the amount of I/O that it performs. Depending on your configuration, this action may affect overall database performance.

To multiplex the online redo log:

1. In EM Express, from the **Storage** menu, select **Redo Log Groups**.

   The Redo Log Groups page appears.

2. Select a group and click **Add Member**.

   The Add Member page appears.

3. In the **File Directory** field, enter the directory where you want the data file to be stored on disk.

   You can create this file in the same directory as the other member of the redo log file group, but it is recommended that you store members on separate disk drives. That way, if there is a drive failure, then you still have access to one member.

4. In the **File Name** field, enter a file name for the new redo log member.

   For example, if your existing member file name is `REDO01.log`, then you might name this member `REDO01a.log`.

5. Click **OK**.

   A confirmation message appears.

6. Repeat Step 2 through Step 5 for every existing log group.

Switching a Log File

When a log switch occurs, the log writer (LGWR) process stops writing to the current online redo log group and starts writing to the next available redo log group. After a log switch, the current online redo log group becomes inactive, and the next available online redo log group becomes the current online redo log group. You can switch a log file using Oracle Enterprise Manager Database Express (EM Express).

You can force a log switch to make the current redo group inactive and available for redo log maintenance operations. Forcing a log switch is useful in configurations with large redo log files that take a long time to fill. For example, you might want to:

- Drop the current redo group, but are not able to do so until the group is inactive
- Archive the current online redo log group members immediately, even though they are not yet completely filled

To switch a log file:

1. In EM Express, from the **Storage** menu, select **Redo Log Groups**.

   The Redo Log Groups page appears.

2. From the **Actions** list, select **Switch Logfile**, and then click **OK**.

   A confirmation message appears.

3. A confirmation message appears. Click **Yes**.

   The status of the group that had been Current changes to Active, and the status of the next group in the list changes from Active to Current.

Managing Undo Data

Beginning with Oracle Database 11*g*, for a default installation, Oracle Database automatically manages the undo data. There is typically no need for database administrator (DBA) intervention. However, if your installation uses Oracle Flashback operations, then you may have to perform some undo management tasks to ensure the success of these operations.

This section provides background information and instructions for managing undo data. It contains the following topics:

- About Undo Data
- About Managing Undo Data
- Viewing Undo Information
- Computing the Minimum Undo Tablespace Size Using the Undo Advisor
- Changing the Undo Tablespace to a Fixed Size

- Changing the Datafiles for an Undo Tablespace to a Fixed Size
- Changing Undo Management Analysis Parameters
- Switching Undo Tablespaces

About Undo Data

When a transaction modifies data, Oracle Database copies the original data before modifying it. The original copy of the modified data is called **undo data**. Saving this information is necessary for the following reasons:

- To undo any uncommitted changes made to the database if a rollback is necessary. A rollback can be needed because a user wants to undo the changes of a misguided or unintentional transaction, or it can be part of a recovery operation.
- To provide **read consistency**, which means that each user can get a consistent view of data, even while other changes may be occurring against the data. With read consistency, a user session does not see uncommitted changes made in other user sessions (sometimes referred to as *dirty reads*). For example, if a user issues a query at 10:00 a.m. and the query lasts for 15 minutes, then the query results reflect the entire state of the data at 10:00 a.m., regardless of update or insert operations performed by other users after the query started.
- To enable certain Oracle Flashback features, such as Oracle Flashback Query and Oracle Flashback Table, which enable you to view or recover data to a previous point in time.

Undo Tablespace

With automatic undo management, undo data is stored in an undo tablespace. Undo tablespaces have additional properties beyond those of permanent tablespaces. There can be multiple undo tablespaces, but only one can be active for an Oracle instance.

When you create the database using Database Configuration Assistant (DBCA), it creates an autoextending undo tablespace named `UNDOTBS1`, with a maximum extension size of 32,767 MB.

Undo Retention

Oracle Database automatically ensures that undo data that is in use by an active transaction is never overwritten until that transaction has been committed. After the transaction has been committed, the space occupied by that undo data can be reused, or overwritten. In this case, that undo data could be overwritten if space in the undo tablespace becomes scarce.

Even after a transaction has been committed, it is useful to retain (not overwrite) its undo data, to ensure the success of Oracle Flashback features and for read consistency for long-running queries. To this end, the database maintains and automatically tunes an **undo retention period**. Committed undo data whose age is less than the undo retention period is retained for use by queries or Oracle Flashback operations.

See Also:

- *Oracle Database Concepts* for more information about read consistency
- *Oracle Database Development Guide* for more information about Oracle Flashback features

---

About Managing Undo Data

Although by default Oracle Database manages undo data and the undo tablespace automatically, if your installation uses Oracle Flashback features, then you may have to perform some undo management tasks to ensure the success of these operations.

Oracle Flashback operations resulting in `snapshot too old` errors indicate that you must intervene to ensure that sufficient undo data is retained to support these operations.

The following methods better support Oracle Flashback operations:

- Set the minimum undo retention period for the autoextending tablespace to be as long as the longest expected Oracle Flashback operation.

  You achieve this goal by setting the `UNDO_RETENTION` initialization parameter. See *Oracle Database Administrator's Guide* for details.

- Change the undo tablespace to a fixed size.

  For an autoextending undo tablespace, Oracle Database always automatically tunes the undo retention period to be slightly longer than the longest-running active query. However, this autotuned retention period may be insufficient to accommodate Oracle Flashback operations. If the undo tablespace has autoextending disabled, or has a fixed size, then Oracle Database uses a different method for tuning the undo retention period to better accommodate Oracle Flashback operations.

  To change the undo tablespace to a fixed size, you must choose a tablespace size that is sufficiently large. If you choose an undo tablespace size that is too small, then the following errors could occur:

  - DML could fail because there is not enough space to accommodate undo data for new transactions.
  - Long-running queries could fail with a `snapshot too old` error, which means that there was insufficient undo data for read consistency.

  Oracle Enterprise Manager Database Express (EM Express) includes an Undo Advisor to help you determine the minimum size for the fixed size of the undo tablespace. See "Computing the Minimum Undo Tablespace Size Using the Undo Advisor".

See Also:

- "About Undo Data"
- "Computing the Minimum Undo Tablespace Size Using the Undo Advisor"
- "Changing the Undo Tablespace to a Fixed Size"

Viewing Undo Information

You can use the Undo Management Details page in Oracle Enterprise Manager Database Express (EM Express) to view the following information about your undo configuration:

- Name and current size of the undo tablespace
- Auto extensible tablespace setting (Yes or No)
- Current undo retention period

Note:

In a multitenant container database (CDB), the EM Express undo management features are available only in the root, not in the pluggable databases (PDBs).

To view undo information:

1. In EM Express, from the **Storage** menu, select **Undo Management**.

   The Undo Management Details page appears.

2. View the undo management information on the Undo Management Details page.

See Also:

"About Undo Data"

Computing the Minimum Undo Tablespace Size Using the Undo Advisor

If you must change the undo tablespace to a fixed size, then use the Undo Advisor in Oracle Enterprise Manager Database Express (EM Express) to help determine the minimum required size. You can also use the Undo Advisor to set the minimum undo retention period.

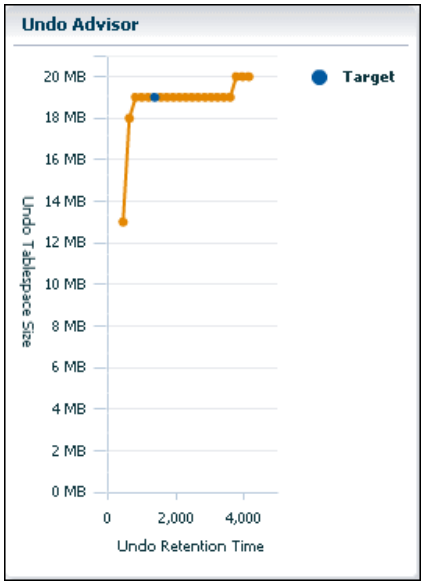
To configure the undo tablespace to have a fixed size, Oracle suggests that you first allow enough time after database creation to run a full workload, thus allowing the undo tablespace to grow to its minimum required size to handle the workload. Then, you can use the Undo Advisor to determine the best size to configure the undo tablespace to allow for future long-running queries and Oracle Flashback operations.

To compute the minimum undo tablespace size using the Undo Advisor:

1. In EM Express, go to the Undo Management Details page.

   See "Viewing Undo Information".

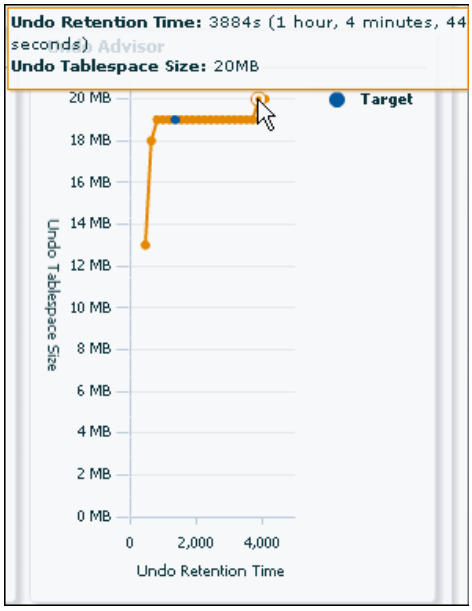
2. In the Configuration section, view the Undo Advisor graph.



   The Undo Advisor displays an orange line. There are several tick-marks, or points, that you can click on the orange line. The blue point on the orange line marks the target undo retention time that is currently set for the database.

3. To determine the recommended minimum size of the undo tablespace for a particular undo retention time, select the point on the orange line closest to the desired undo retention time.

   In the following figure, for example, the point on the orange line for the undo retention time of 3884 seconds has been clicked:



The Undo Advisor shows its recommendation that the undo tablespace size should be at least 20 MB for this undo retention time of 3884 seconds.

Note:

Using the Undo Advisor to gets its recommendation for the required minimum tablespace size for a particular undo retention time does not change the size of the undo tablespace. You must follow the instructions in "Changing the Undo Tablespace to a Fixed Size" to change the size of the undo tablespace.

See Also:

- "About Undo Data"
- "About Managing Undo Data"

Changing the Undo Tablespace to a Fixed Size

You change the undo tablespace to a fixed size to prevent the tablespace from growing too large or to better support Oracle Flashback operations. You can use Oracle Enterprise Manager Database Express (EM Express) to change the undo tablespace to a fixed size.

Note:

Only bigfile tablespaces can be changed to a fixed size. However, individual datafiles for a smallfile tablespace can be changed to a fixed size. See "Changing the Datafiles for an Undo Tablespace to a Fixed Size" for instructions for changing individual data files for a smallfile tablespace to a fixed size.

To change the undo tablespace to a fixed size:

1. In EM Express, go to the Undo Management Details page, as described in "Viewing Undo Information".
2. After determining the minimum required undo tablespace size, click the link after the **Name** field in the Tablespace section.

   The Tablespace page appears, with the undo tablespace displayed.
3. Select the undo tablespace. Then from the **Actions** menu, select **Resize**.

   The Resize Tablespace page appears.
4. In the **File Size** field, enter the computed minimum size for the undo tablespace.

   See "Computing the Minimum Undo Tablespace Size Using the Undo Advisor".
5. Click **OK**.

   A confirmation message appears.

See Also:

- "About Undo Data"
- "About Managing Undo Data"

Changing the Datafiles for an Undo Tablespace to a Fixed Size

You change the datafiles for an undo tablespace to a fixed size to prevent the tablespace from growing too large or to better support Oracle Flashback operations. You can use Oracle Enterprise Manager Database Express (EM Express) to change the datafiles for an undo tablespace to a fixed size.

To change the datafiles for an undo tablespace to a fixed size:

1. In EM Express, go to the Undo Management Details page, as described in "Viewing Undo Information".
2. After determining the minimum required undo tablespace size, click the link after the **Name** field in the Tablespace section.

   The Tablespace page appears, with the undo tablespace displayed.
3. Select and expand the undo tablespace. Select one of the datafiles for the undo tablespace. Then from the **Actions** menu, select **Resize**.

   The Resize Datafile page appears.
4. In the **Size** field, enter the computed minimum size for the undo tablespace.

   See "Computing the Minimum Undo Tablespace Size Using the Undo Advisor".
5. Click **OK**.

   A confirmation message appears.

6. You can perform these steps for all the datafiles for the undo tablespace to change them all to a fixed size.

Changing Undo Management Analysis Parameters

You can change the current analysis period and the desired undo retention period using Oracle Enterprise Manager Database Express (EM Express).

To change the analysis period and the undo retention period:

1. In EM Express, go to the Undo Management Details page, as described in "Viewing Undo Information".
2. Click **Change Analysis Parameters**.

   The Change Analysis Parameters page appears.
3. To change the analysis period, in the **Analysis Period** field, select the desired analysis period.
4. To change the undo retention period, select either:
   - **Use Required Undo Retention**
   - **Specify Undo Retention in Seconds**

     Enter the number of seconds to use for the retention period in the text box.

Switching Undo Tablespaces

You can switch from one undo tablespace to another using Oracle Enterprise Manager Database Express (EM Express).

To switch the undo tablespace:

1. In EM Express, go to the Undo Management Details page, as described in "Viewing Undo Information".
2. Click **Switch Undo Tablespace**.

   The Switch Undo Tablespace page appears.
3. In the **Switch to Undo Tablespace** field, select the name of the undo tablespace you want to switch to. This field includes the names of available undo tablespaces for the database.

   For example, if the current undo tablespace is named UD1 and you want to switch to the undo tablespace named UD2, select **UD2** in the **Switch to Undo Tablespace** field.
4. Click OK.

   A confirmation message appears.

See Also:

- "About Undo Data"
- "About Managing Undo Data"
- *Oracle Database Administrator's Guide* for more information about switching undo tablespaces

Storage: Oracle by Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter and includes annotated screenshots.

To view the Managing Database Storage Structure OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6286,1
```

7

Administering User Accounts and Security

This chapter describes how to create and manage user accounts using Oracle Enterprise Manager Database Express (EM Express). It contains the following sections:

About User Accounts

For users to access your database, you must create user accounts and grant appropriate database access privileges to those accounts. A user account is identified by a user name and defines the attributes of the user, including the following:

- Authentication method
- Password for database authentication
- Default tablespaces for permanent and temporary data storage
- Tablespace quotas
- Account status (locked or unlocked)
- Password status (expired or not)

When you create a user account, you must not only assign a user name, a password, and default tablespaces for the account, but you must also do the following:

- Grant the appropriate system privileges, object privileges, and roles to the account.
- If the user will be creating database objects, then give the user account a space usage quota on each tablespace in which the objects will be created.

Oracle recommends that you grant each user just enough privileges to perform his job, and no more. For example, a database application developer needs privileges to create and modify tables, indexes, views, and stored procedures, but does not need (and should not be granted) privileges to drop (delete) tablespaces or recover the database. You can create user accounts for database administration, and grant only a subset of administrative privileges to those accounts.

In addition, you may want to create user accounts that are used by applications only. That is, nobody logs in with these accounts; instead, applications use these accounts to connect to the database, and users log in to the applications. This type of user account avoids giving application users the ability to log in to the database directly, where they could unintentionally cause damage. See "About User Privileges and Roles" for more information.

When you create a user account, you are also implicitly creating a schema for that user. A **schema** is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates. The schema name is the same as the user name, and can be used to unambiguously refer to objects owned by the user. For example, `hr.employees` refers to the table named `employees` in the `hr` schema. (The `employees` table is owned by `hr`.) The terms *database object* and *schema object* are used interchangeably.

When you delete a user, you must either simultaneously delete all schema objects of that user, or you must have previously deleted the schema objects in separate operations.

Predefined User Accounts

In addition to the user accounts that you create, the database includes several user accounts that are automatically created upon installation.

All databases include the administrative accounts `SYS`, `SYSTEM`, and `DBSNMP`. **Administrative accounts** are highly privileged accounts, and are needed only by individuals authorized to perform administrative tasks such as starting and stopping the database, managing database memory and storage, creating and managing database users, and so on. You log in to Oracle Enterprise Manager Database Express (EM Express) with `SYS` or `SYSTEM`. You assign the passwords for these accounts when you create the database with Oracle Database Configuration Assistant (DBCA). You must not delete these accounts.

All databases also include **internal accounts**, which are automatically created so that individual Oracle Database features or components such as Oracle Application Express can have their own schemas. To protect these accounts from unauthorized access, they are initially locked and their passwords are expired. (A **locked account** is an account for which login is disabled.) You must not delete internal accounts, and you must not use them to log in to the database.

Your database may also include **sample schemas**, if you chose the option to create the sample schemas in your database when the database was installed. The sample schemas are a set of interlinked schemas that enable Oracle documentation and Oracle instructional materials to illustrate common database tasks. These schemas also provide a way for you to experiment without endangering production data.

Each sample schema has a user account associated with it. For example, the `hr` user account owns the `hr` schema, which contains a set of simple tables for a human resources application. The sample schema accounts are also initially locked and have an expired password. As the database administrator, you are responsible for unlocking these accounts and assigning passwords to these accounts.

See Also:

- *Oracle Database 2 Day + Security Guide* for a list of predefined user accounts
- "Locking and Unlocking User Accounts"
- "About Administrative Accounts and Privileges"
- "SYS and SYSTEM Users" for information about the recommended alternative to using the `SYSTEM` account for day-to-day administrative tasks
- "Administering Database User Accounts"
- *Oracle Database Sample Schemas* for a description of the sample schemas
- *Oracle Database Concepts* for an overview of database security

About Commonality in a CDB

In a multitenant container database (CDB), the basic principle of commonality is that *a common phenomenon is the same in every existing and future container*. In a CDB, "common" means "common to all containers." In contrast, a local phenomenon is restricted to exactly one existing container.

A corollary to the principle of commonality is that *only a common user can alter the existence of common phenomena*. More precisely, only a common user connected to the root can create, destroy, or modify CDB-wide attributes of a common user or role.

See Also:

Oracle Database Concepts and *Oracle Database Administrator's Guide* for more information about CDBs and pluggable databases (PDBs)

Common Users in a CDB

A common user is a database user that has the same identity in the root and in every existing and future pluggable database (PDB). Every common user can connect to and perform operations within the root, and within any PDB in which it has privileges.

Every common user is either Oracle-supplied or user-created. Examples of Oracle-supplied common users are `SYS` and `SYSTEM`.

Common users have the following characteristics:

- A common user can log in to any container (including `CDB$ROOT`) in which it has the `CREATE SESSION` privilege.

  A common user need not have the same privileges in every container. For example, the `c##dba` user may have the privilege to create a session in the root and in one PDB, but not to create a session in a different PDB. Because a common user with the appropriate privileges can switch between containers, a common user in the root can administer PDBs.

- The name of every user-created common user must begin with the characters `c##` or `C##`. (Oracle-supplied common user names do not have this restriction.)

No local user name may begin with the characters `c##` or `C##`.

- The names of common users must contain only ASCII or EBCDIC characters.
- Every common user is uniquely named across all containers.

  A common user resides in the root, but must be able to connect to every PDB with the same identity.
- The schemas for a common user can differ in each container.

  For example, if `c##dba` is a common user that has privileges on multiple containers, then the `c##dba` schema in each of these containers may contain different objects.

See Also:

- *Oracle Database Concepts* for more information about common users in a multitenant container database (CDB)
- *Oracle Database Security Guide* to learn about common and local account

Local Users in a CDB

A local user is a user that is not common and that can operate only within a single pluggable database (PDB). Local users have the following characteristics:

- A local user is specific to a particular PDB and owns a schema in this PDB.
- A local user cannot be created in the root.
- A local user on one PDB cannot log in to another PDB or to the root.
- The name of a local user cannot begin with the characters `c##` or `C##`.
- The name of a local user must only be unique within its PDB.
- The user name and the PDB in which that user schema is contained determine a unique local user. For example, a local user and schema named `rep` can exist on a PDB named `hrpdb`. A completely independent local user and schema named `rep` can exist on a PDB named `salespdb`.
- Whether local users can access objects in a common schema depends on their user privileges.

  For example, the `c##dba` common user may create a table in the `c##dba` schema on the `hrpdb` PDB. Unless c##dba grants the necessary privileges to the local `hr` user on this table, `hr` cannot access it.

See Also:

- *Oracle Database Concepts* for more information about local users
- *Oracle Database Concepts* for a scenario involving local users in two PDBs
- *Oracle Database Security Guide* to learn about local accounts

About User Privileges and Roles

User privileges provide a basic level of database security. They are designed to control user access to data and to limit the kinds of SQL statements that users can execute. When creating a user, you grant privileges to enable the user to connect to the database, to run queries and make updates, to create schema objects, and more.

The main types of user privileges are as follows:

- **System privileges**—A system privilege gives a user the ability to perform a particular action, or to perform an action on any schema objects of a particular type. For example, the system privilege `CREATE TABLE` permits a user to create tables in the schema associated with that user, and the system privilege `CREATE USER` permits a user to create database users.
- **Object privileges**—An objectprivilege gives a user the ability to perform a particular action on a specific schema object. Different object privileges are available for different types of schema objects. The privilege to select rows from the `EMPLOYEES` table or to delete rows from the `DEPARTMENTS` table are examples of object privileges.

Managing privileges is made easier by using **roles**, which are named groups of related privileges. You create roles, grant system and object privileges to the roles, and then grant roles to users. You can also grant roles to other roles. Unlike schema objects, roles are not contained in any schema.

Table 1 lists three widely used roles that are predefined in Oracle Database. You can grant these roles when you create a user or at any time thereafter.

Table 1 Oracle Database Predefined Roles

Role Name	Description
`CONNECT`	Enables a user to connect to the database. Grant this role to any user or application that needs database access.
`RESOURCE`	Enables a user to create, modify, and delete certain types of schema objects in the schema associated with that user. Grant this role only to developers and to other users that must create schema objects. This role grants a subset of the create object system privileges. For example, it grants the `CREATE TABLE` system privilege, but does not grant the `CREATE VIEW` system privilege. It grants only the following privileges: `CREATE CLUSTER`, `CREATE INDEXTYPE`, `CREATE OPERATOR`, `CREATE PROCEDURE`, `CREATE SEQUENCE`, `CREATE TABLE`, `CREATE TRIGGER`, `CREATE TYPE`.
`DBA`	Enables a user to perform most administrative functions, including creating users and granting privileges; creating and granting roles; creating, modifying, and deleting schema objects in any schema; and more. It grants all system privileges, but does not include the privileges to start or shut down the database instance. It is by default granted to users `SYS` and `SYSTEM`.

See Also:

- "Administering Roles"
- "Administering Database User Accounts"
- Managing Schema Objects
- *Oracle Database 2 Day + Security Guide* for more information about privileges and roles
- *Oracle Database SQL Language Reference* for tables of system privileges, object privileges, and predefined roles
- *Oracle Database Concepts* for an overview of database security

About Common and Local Roles in a CDB

Every Oracle-supplied role is common. In Oracle-supplied scripts, every privilege or role granted to Oracle-supplied users and roles is granted commonly, with one exception: system privileges are granted locally to the common role `PUBLIC`. User-created roles are either local or common.

See Also:

Oracle Database Concepts for more information about grants to `PUBLIC` in a multitenant container database (CDB)

Common Roles in a CDB

A common role is a database role that exists in the root and in every existing and future pluggable database (PDB). Common roles are useful for cross-container operations, ensuring that a common user has a role in every container.

Every common role is either user-created or Oracle-supplied. All Oracle-supplied roles are common, such as `DBA` and `PUBLIC`. User-created common roles must have names starting with `C##` or `c##`, and must contain only ASCII or EBCDIC characters. For example, a multitenant container database (CDB) administrator might create common user `c##dba`, and then grant the DBA role commonly to this user, so that `c##dba` has the DBA role in any existing and future PDB.

A user can only perform common operations on a common role, for example, granting privileges commonly to the role, when the following criteria are met:

- The user is a common user whose current container is root.
- The user has the `SET CONTAINER` privilege granted commonly, which means that the privilege applies in all containers.
- The user has privilege controlling the ability to perform the specified operation, and this privilege has been granted commonly.

For example, to create a common role, a common user must have the `CREATE ROLE` and the `SET CONTAINER` privileges granted commonly. Common roles created using Oracle Enterprise Manager Database Express (EM Express) must be created in the root.

See Also:

- *Oracle Database Concepts* for more details about roles and privileges granted commonly in a CDB
- *Oracle Database Security Guide* to learn how to manage common roles

Local Roles in a PDB

A local role exists only in a single pluggable database (PDB), just as a role in a non-CDB exists only in the non-CDB. A local role can only contain roles and privileges that apply within the container in which the role exists.

PDBs in the same multitenant container database (CDB) may contain local roles with the same name. For example, the user-created role `pdbadmin` may exist in both the `hrpdb` and `salespdb` PDBs. These roles are completely independent of each other, just as they would be in separate non-CDBs.

A local role created using Oracle Enterprise Manager Database Express (EM Express) must be created in the PDB where it will be used.

See Also:

Oracle Database Security Guide to learn how to manage local roles

About Privilege and Role Grants in a CDB

Just as in a non-CDB, users in a multitenant container database (CDB) can grant roles and privileges. A key difference in a CDB is the distinction between roles and privileges that are locally granted and commonly granted. A privilege or role granted locally is exercisable only in the container in which it was granted. A privilege or role granted commonly is exercisable in every existing and future container.

Users and roles may be common or local. However, a privilege is in itself neither common nor local. If a user grants a privilege locally using the `CONTAINER=CURRENT` clause, then the grantee has a privilege exercisable only in the current container. If a user grants a privilege commonly using the `CONTAINER=ALL` clause, then the grantee has a privilege exercisable in any existing and future container.

Note:

When you use Oracle Enterprise Manager Database Express (EM Express) to grant privilege or roles in a CDB, the container in which the privilege is granted determines whether it is a commonly granted or locally granted privilege or role.

For example, when you use EM Express to grant a privilege in the root, the privilege is a commonly granted privilege that the grantee can exercise in any existing and future container. When you use EM Express to grant a privilege in a pluggable database (PDB), the privilege is a locally granted privilege that the grantee can exercise only in that PDB.

In a CDB, every act of granting, whether local or common, occurs within a specific container. The basic principles of granting are as follows:

- Both common and local phenomena may grant and be granted locally.
- Only common phenomena may grant or be granted commonly.

Local users, roles, and privileges are by definition restricted to a particular container. Thus, local users may not grant roles and privileges commonly, and local roles and privileges may not be granted commonly.

See Also:

Oracle Database Concepts for more details about these granting principles

Privileges and Roles Granted Commonly in a CDB

Privileges and common roles may be granted commonly. According to the principles of granting in a pluggable database (PDB), users or roles may be granted roles and privileges commonly only if the grantees and grantors are both *common*; and if a role is being granted commonly, then the role itself must be common.

See Also:

- *Oracle Database Concepts* for more information about privileges and roles granted commonly in a multitenant container database (CDB), and for a table that shows what phenomena can be granted commonly
- *Oracle Database Concepts* for a detailed scenario of granting roles and privileges commonly and locally in a CDB

Privileges and Roles Granted Locally in a CDB

Roles and privileges may be granted locally to users and roles *regardless* of whether the grantees, grantors, or roles being granted are local or common.

See Also:

- *Oracle Database Concepts* for more information about privileges and roles granted locally in a multitenant container database (CDB), and for a table that shows what phenomena can be granted locally
- *Oracle Database Concepts* for a detailed scenario of granting roles and privileges commonly and locally in a CDB

About Administrative Accounts and Privileges

Administrative accounts and privileges enable you to perform administrative functions such as managing users, managing database memory, and starting up and shutting down the database.

This section contains the following topics:

- SYS and SYSTEM Users
- SYSDBA and SYSOPER System Privileges

See Also:

- "About User Accounts"
- "About User Privileges and Roles"
- "Administering Database User Accounts"

SYS and SYSTEM Users

The following administrative user accounts are automatically created when you install Oracle Database. They are both created with the password that you supplied upon installation, and they are both automatically granted the DBA role.

- SYS

  This account can perform all administrative functions. All base (underlying) tables and views for the database data dictionary are stored in the SYS schema. These base tables and views are critical for the operation of Oracle Database. To maintain the integrity of the data dictionary, tables in the SYS schema are manipulated only by the database. They should never be modified by any user or database administrator. You must not create any tables in the SYS schema.

  The SYS user is granted the SYSDBA privilege, which enables a user to perform high-level administrative tasks such as backup and recovery.

- SYSTEM

  This account can perform all administrative functions except the following:

  - Backup and recovery
  - Database upgrade

  While this account can be used to perform day-to-day administrative tasks, Oracle strongly recommends creating named user accounts for administering the Oracle database to enable monitoring of database activity.

Note:

SYSBACKUP is another automatically created account that is used to perform backup and recovery. See "Configuring Users to Perform Backup and Recovery" for more information.

SYSDBA and SYSOPER System Privileges

SYSDBA and SYSOPER are administrative privileges required to perform high-level administrative operations such as creating, starting up, shutting down, backing up, or recovering the database. The SYSDBA system privilege is for fully empowered database administrators and the SYSOPER system privilege allows a user to perform basic operational tasks, but without the ability to look at user data.

The SYSDBA and SYSOPER system privileges allow access to a database instance even when the database is not open. Control of these privileges is therefore completely outside of the database itself. This control enables an administrator who is granted one of these privileges to connect to the database instance to start the database.

You can also think of the SYSDBA and SYSOPER privileges as types of connections that enable you to perform certain database operations for which privileges cannot be granted in any other way. For example, if you have the SYSDBA privilege, then you can connect to the database using AS SYSDBA.

The SYS user is automatically granted the SYSDBA privilege upon installation. When you log in as user SYS, you must connect to the database as SYSDBA or SYSOPER. Connecting as a SYSDBA user invokes the SYSDBA privilege; connecting as SYSOPER invokes the SYSOPER privilege. EM Express allows you to log in as user SYS and connect as SYSDBA or SYSOPER.

When you connect with the SYSDBA or SYSOPER privilege, you connect with a default schema, not with the schema that is generally associated with your user name. For SYSDBA this schema is SYS; for SYSOPER the schema is PUBLIC.

Note:

When you connect as user SYS, you have unlimited privileges on data dictionary tables. Be certain that you do not modify any data dictionary tables.

See Also:

Oracle Database Administrator's Guide for the operations authorized with the SYSDBA and SYSOPER privileges

Administering Roles

Roles are named groups of related system and object privileges. You create roles and then assign them to users and to other roles.

This section contains the following topics:

- Viewing Roles
- Example: Creating a Role
- Example: Modifying a Role
- Deleting a Role

See Also:

- "About User Privileges and Roles"
- *Oracle Database 2 Day + Security Guide* for more information about administering user security, roles, and privileges

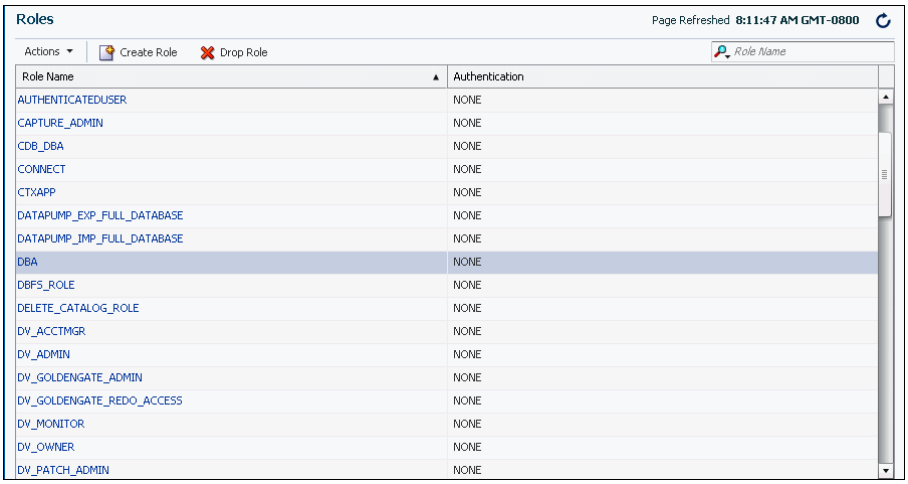
Viewing Roles

You view roles on the Roles page of Oracle Enterprise Manager Database Express (EM Express).

To view roles:

1. Log into EM Express with a user account that has privileges to manage roles. An example of such a user account is SYSTEM.
2. From the **Security** menu, select **Roles**.

   The Roles page appears.



3. To view the details of a particular role, select the name of the role you want to view, and then from the **Actions** list, select **View Details**.

   You can also use the Search area of the page to search for a particular role. In the Search field, enter the first few letters of the role. As you type, the list of roles in the table are restricted to the roles whose names include the letters you entered.

   The View Role page appears. In this page, you can see all the privileges and roles granted to the selected role.

See Also:

"SYS and SYSTEM Users" for information about the recommended alternative to using the SYSTEM account for day-to-day administrative tasks

---

Example: Creating a Role

You can use Oracle Enterprise Manager Database Express (EM Express) to create a role called APPDEV for application developers in a pluggable database (PDB). Because application developers must be able to create, modify, and delete the schema objects that their applications use, you want the APPDEV role to include the system privileges shown in Table 2 .

Table 2 System Privileges Granted to the APPDEV Role

Privilege	Description
CREATE TABLE	Enables a user to create, modify, and delete tables in his schema.
CREATE VIEW	Enables a user to create, modify, and delete views in his schema.
CREATE PROCEDURE	Enables a user to create, modify, and delete procedures in his schema.
CREATE TRIGGER	Enables a user to create, modify, and delete triggers in his schema.
CREATE SEQUENCE	Enables a user to create, modify, and delete sequences in his schema.
CREATE SYNONYM	Enables a user to create, modify, and delete synonyms in his schema.

Note:

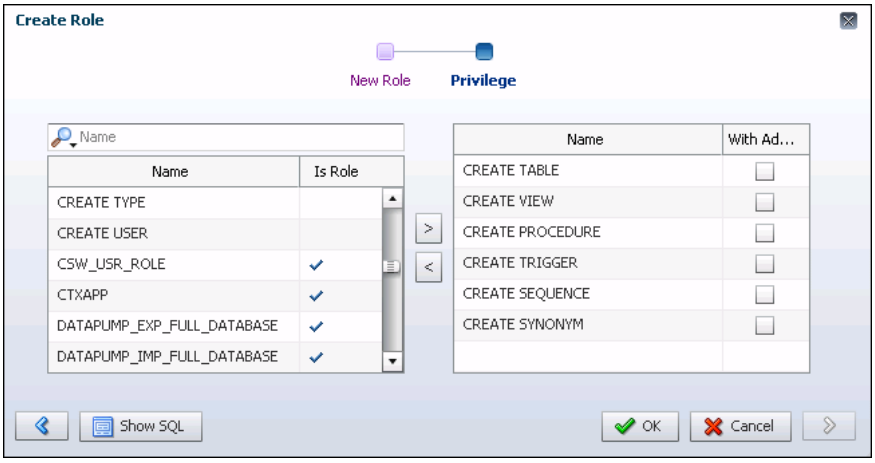
If you create an APPDEV role for application developers at your company, you should follow the principle of least privilege, in which you grant to your application developers only the privileges needed to perform their job function, and no more. Therefore, the set of privileges that you grant to the APPDEV role for your company may be different than the system privileges that are granted to the APPDEV role in Table 2 .

To create the APPDEV role:

1. In EM Express, go to the Roles page, as described in "Viewing Roles".
2. Click **Create Role**.

   The Create Role wizard appears, with the New Role page showing.
3. In the **Role Name** field, enter APPDEV.
4. Click the right arrow.

   The Privilege page appears.



The available system privileges and roles table on the left shows the available system privileges that can be assigned to the role. Roles are identified by a check mark in the **Is Role** column. The selected system privileges and roles table on the right shows the system privileges and roles that are currently selected for the role.

Select one or more system privileges or roles in either table, and then click the appropriate arrow button to move those privileges to the other table.

Move the CREATE TABLE, CREATE VIEW, CREATE PROCEDURE, CREATE TRIGGER, CREATE SEQUENCE, and CREATE SYNONYM system privileges to the selected system privileges and roles table for the APPDEV role that is being created.

In the selected system privileges and roles table, enable the **WITH ADMIN** option for a system privilege or role if you want users who will be granted the APPDEV role you are defining to be able to grant the system privilege or role in the selected system privileges and roles table to other users.

5. Click **OK**.

   The APPDEV role now appears in the table of database roles on the Roles page.

Video:

Oracle Database 12c: Creating a Role Using Oracle Enterprise Manager Database Express

---

Example: Modifying a Role

You can modify a role using Oracle Enterprise Manager Database Express (EM Express). For example, suppose your applications make use of Oracle Streams Advanced Queuing, and you determine that developers must be granted the roles AQ_ADMINISTRATOR_ROLE and AQ_USER_ROLE to develop and test their applications. You can edit the APPDEV role to grant it these two Advanced Queuing roles.

To modify the APPDEV role:

1. In EM Express, go to the Roles page, as described in "Viewing Roles".
2. Select the APPDEV role, and from the **Actions** menu, choose **Alter Privileges**.

   The Alter Privileges page appears.
3. Move the AQ_ADMINISTRATOR_ROLE and AQ_USER_ROLE roles from the available system privileges and roles table on the left to the selected system privileges and roles table on the right to grant these two roles to the APPDEV role.
4. Click **OK**.

   A confirmation statement appears.

Deleting a Role

You can delete a role using Oracle Enterprise Manager Database Express (EM Express). Use caution when deleting a role, because EM Express deletes a role even if that role is currently granted to one or more users. Dropping (deleting) a role automatically removes the privileges associated with that role from all users that had been granted the role.

To delete a role:

1. In EM Express, go to the Roles page, as described in "Viewing Roles".
2. Select the role you want to delete, and then click **Drop Role**.

   A confirmation page appears.
3. Click **OK**.

   A confirmation message indicates that the role has been deleted successfully.

Administering Database User Accounts

This section provides instructions for using Oracle Enterprise Manager Database Express (EM Express) to create and manage user accounts for the people and applications that use your database. It contains the following topics:

- Viewing User Accounts
- Example: Creating a User Account
- Creating a New User Account by Duplicating an Existing User Account
- Example: Granting Privileges and Roles to a User Account
- Example: Assigning a Tablespace Quota to a User Account
- Example: Modifying a User Account
- Locking and Unlocking User Accounts
- Expiring a User Password
- Example: Deleting a User Account

See Also:

"About User Accounts"

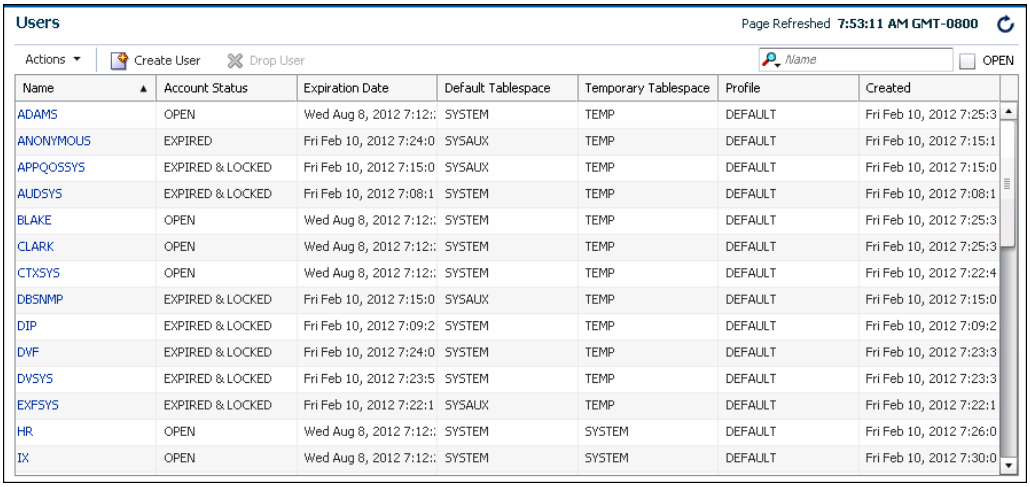
Viewing User Accounts

You view user accounts on the Users page of Oracle Enterprise Manager Database Express (EM Express).

To view users:

1. Log into EM Express with a user account that has privileges to manage users, for example, `SYSTEM`.
2. From the **Security** menu, select **Users**.

   The Users page appears. In a multitenant container database (CDB), this page is named the Common Users page.



Name	Account Status	Expiration Date	Default Tablespace	Temporary Tablespace	Profile	Created
ADAMS	OPEN	Wed Aug 8, 2012 7:12::	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:25:3
ANONYMOUS	EXPIRED	Fri Feb 10, 2012 7:24:0	SYSAUX	TEMP	DEFAULT	Fri Feb 10, 2012 7:15:1
APPQOSSYS	EXPIRED & LOCKED	Fri Feb 10, 2012 7:15:0	SYSAUX	TEMP	DEFAULT	Fri Feb 10, 2012 7:15:0
AUDSYS	EXPIRED & LOCKED	Fri Feb 10, 2012 7:08:1	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:08:1
BLAKE	OPEN	Wed Aug 8, 2012 7:12::	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:25:3
CLARK	OPEN	Wed Aug 8, 2012 7:12::	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:25:3
CTXSYS	OPEN	Wed Aug 8, 2012 7:12::	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:22:4
DBSNMP	EXPIRED & LOCKED	Fri Feb 10, 2012 7:15:0	SYSAUX	TEMP	DEFAULT	Fri Feb 10, 2012 7:15:0
DIP	EXPIRED & LOCKED	Fri Feb 10, 2012 7:09:2	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:09:2
DVF	EXPIRED & LOCKED	Fri Feb 10, 2012 7:24:0	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:23:3
DVSYS	EXPIRED & LOCKED	Fri Feb 10, 2012 7:23:5	SYSTEM	TEMP	DEFAULT	Fri Feb 10, 2012 7:23:3
EXFSYS	EXPIRED & LOCKED	Fri Feb 10, 2012 7:22:1	SYSAUX	TEMP	DEFAULT	Fri Feb 10, 2012 7:22:1
HR	OPEN	Wed Aug 8, 2012 7:12::	SYSTEM	SYSTEM	DEFAULT	Fri Feb 10, 2012 7:26:0
IX	OPEN	Wed Aug 8, 2012 7:12::	SYSTEM	SYSTEM	DEFAULT	Fri Feb 10, 2012 7:30:0

3. To view the details of a particular user, do one of the following:

   - Click the user name.
   - Select the user by clicking anywhere in the row except on the user name, and from the **Actions** menu, select **View Details**.

   The View User page appears, and displays all user attributes.

See Also:

"SYS and SYSTEM Users" for information about the recommended alternative to using the `SYSTEM` account for day-to-day administrative tasks

Example: Creating a User Account

You can use Oracle Enterprise Manager Database Express (EM Express) to create a user account in a pluggable database (PDB) for a database application developer named Nick. Because Nick is a developer, you want to grant him the database privileges and roles that he requires to build and test his applications. You also want to give Nick a 16 MB quota on his default tablespace so that he can create schema objects in that tablespace.

To create the user Nick:

1. In EM Express, go to the Users page, as described in "Viewing User Accounts".
2. On the Users page, click **Create User**.

   The Create User wizard appears, showing the User Account page.
3. Enter the following values:

   - In the **Name** field, enter `NICK`.
   - Accept the default value `Password` in the Authentication list.

     For information about other more advanced methods to authenticate users, see *Oracle Database Security Guide*.

- In the **Password** and **Confirm Password** fields, enter a secure password for user Nick.

  See *Oracle Database Security Guide* for more information about secure passwords.

- In the Profile list, accept the value `DEFAULT`.

  This setting assigns the default password policy to user Nick.

  See "Setting the Database Password Policy".

- Enable the **Password Expired** option. When this option is enabled at user creation time, then the user must create a new password the first time he logs into his account.

- Do not select **Account Locked**.

  You can lock the user account later to prevent users from logging in with it. To temporarily deny access to a user account, locking the user account is preferable to deleting it, because deleting it also deletes all schema objects owned by the user.

4. Click the right arrow button.

   The Tablespaces page appears.

5. Enter the following values:

   - For the **Default Tablespace** field, select the **USERS** tablespace.

     All schema objects that Nick creates will then be created in the `USERS` tablespace unless he specifies otherwise. If you leave the Default Tablespace field blank, then Nick is assigned the default tablespace for the database, which is `USERS` in a newly installed database. For more information about the `USERS` tablespace, see "About Tablespaces".

   - For the **Temporary Tablespace** field, select the **TEMP** tablespace.

     If you leave the **Temporary Tablespace** field blank, then Nick is assigned the default temporary tablespace for the database, which is `TEMP` in a newly installed database. For more information about the `TEMP` tablespace, see "About Tablespaces".

6. Click the right arrow button.

   The Privilege page appears.

7. Grant roles, system privileges, and object privileges to the user, as described in "Example: Granting Privileges and Roles to a User Account".

8. Assign a 16 MB quota on the `USERS` tablespace, as described in "Example: Assigning a Tablespace Quota to a User Account".

See Also:

Oracle Database 2 Day + Security Guide

Video:

Oracle Database 12c: Creating a User using Oracle Enterprise Manager Database Express

Creating a New User Account by Duplicating an Existing User Account

To create a user account that is similar in attributes to an existing user account, you can duplicate the existing user account. You can use Oracle Enterprise Manager Database Express (EM Express) to create a new user account by duplicating an existing user account.

To create a new user account by duplicating an existing user account:

1. In EM Express, go to the Users page, as described in "Viewing User Accounts".
2. Select a user to duplicate.
3. Click **Create Like**.
4. The Create User wizard appears, showing the User Account page.

   To finish creating the new user, follow steps 3 through 8 in "Example: Creating a User Account".

Example: Granting Privileges and Roles to a User Account

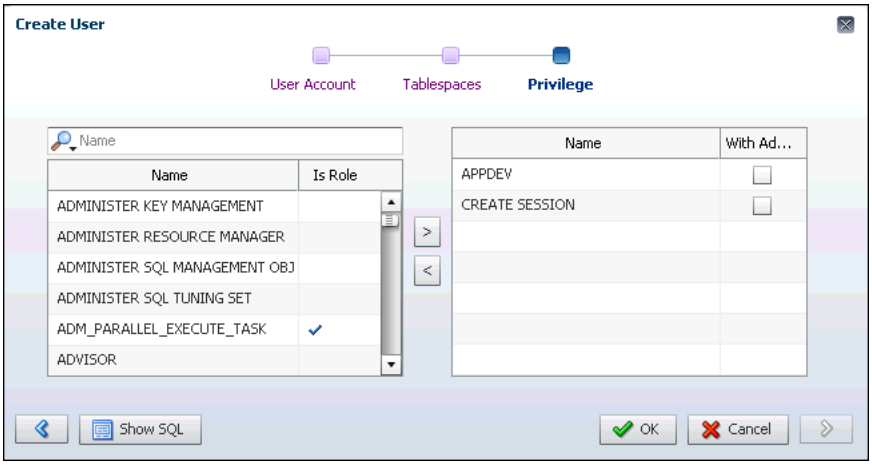
You can use Oracle Enterprise Manager Database Express (EM Express) to grant privileges and roles to a user account. For example, suppose you are creating or modifying a user account named `NICK` for an application developer named Nick. Because Nick is a database application developer, you will grant `NICK` the `APPDEV` role, which enables him to create database objects in his own schema (you created the `APPDEV` role in "Example: Creating a Role"). You also want him to be able to connect to the database, so you will grant him the `CREATE SESSION` system privilege. In addition, because he is developing a human resources application, you want him to be able to view the tables in the `hr` sample schema that is provided with Oracle Database, so you will grant him the `SELECT` object privilege for all the tables in the `hr` sample schema. The sample schemas that are provided by Oracle Database include fictitious data that is intended to be used for example and demonstration purposes, so granting `NICK` access to the `hr` sample schema provided by Oracle Database does not grant him access to any sensitive data. The following table summarizes the privileges and roles that will be granted to `NICK`.

Grant Type	Privilege or Role Name
Role	`APPDEV`
System privilege	`CREATE SESSION`
Object privilege	`SELECT` on all tables in the `hr` sample schema provided with Oracle Database

The following example assumes that you are in the process of creating the user account for Nick. The example also assumes that you have not yet granted any privileges or roles to Nick.

To grant privileges and roles to the user Nick:

1. On the Privilege page in EM Express, find and select the `APPDEV` role and the `CREATE SESSION` system privilege in the available system privileges and roles table on the left, and use the right arrow button to move them to the selected system privileges and roles table on the right.



2. Click **OK**.

   A confirmation message appears, and user NICK is created.

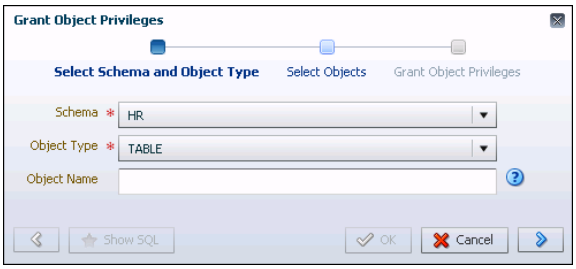
3. Go to the View User page for user NICK, as described in "Viewing User Accounts".

4. Click the Object Privileges subtab.

   The Object Privileges subpage appears.

5. Click the **Grant** button.

   The Grant Object Privileges wizard appears, with the Select Schema and Object Type page displayed.



6. In the **Schema** list, select **HR**, and in the **Object Type** list, select **Tables**.

   In this example, user `NICK` is being granted the `SELECT` object privilege for all the tables in the `hr` sample schema provided by Oracle Database, which contains fictitious data intended for example and demonstration purposes. He is not being granted access to any sensitive data.

7. Click the right arrow button.

   The Select Objects page appears.

8. Move all the tables from the available objects table on the left to the selected objects table on the right to make those tables available to user NICK.
9. Click the right arrow button.

   The Grant Object Privileges page appears.
10. Select the `SELECT` privilege in the Privilege list to grant `NICK` the `SELECT` privilege for all of the tables in the `hr` sample schema provided by Oracle Database.
11. Click **OK** to save the new object privilege grants.

    A confirmation message appears.

See Also:

- "About User Privileges and Roles"
- *Oracle Database 2 Day + Security Guide*

Example: Assigning a Tablespace Quota to a User Account

Suppose you are creating or modifying a user account named Nick. You can assign Nick a space usage quota of 16 MB on his default tablespace using Oracle Enterprise Manager Database Express (EM Express).

You must assign Nick a tablespace quota on his default tablespace before he can create objects in that tablespace. (This is also true for any other tablespace in which Nick wants to create objects.) After a quota is assigned to Nick for a particular tablespace, the total space used by all of his objects in that tablespace cannot exceed the quota. You can also assign a quota of `UNLIMITED`.

The following example assumes that you are in the process of creating the user account for Nick or editing the account. The example also assumes that Nick has not yet been assigned a quota on any tablespaces.

To assign a tablespace quota to user Nick:

1. In EM Express, go to the View User page for user Nick, as described in "Viewing User Accounts".
2. Select the **Quotas** subpage.

   The Quotas subpage appears, showing that user Nick does not have a quota assigned on any tablespace.
3. Select the `USERS` tablespace, and then click **Edit**.

   The Alter Quota page appears.
4. In the **Quota** field, enter **16M** to assign a quota of 16 MB on the USERS tablespace for user Nick.

   When you enter a value in the **Quota** field, EM Express rounds the value up to a multiple of the number of database blocks when it changes the quota for the selected tablespace. For example, if the database uses database blocks that are 8K in size and you enter a value of 10K in the **Quota** field, EM Express will round 10K up to 16K (2 blocks) when it changes the quota for the tablespace.

5. Click **OK** to save the new quota assignment.

   A confirmation message appears.

Example: Modifying a User Account

You can use Oracle Enterprise Manager Database Express (EM Express) to remove the quota limitations for the user Nick on his default tablespace, USERS. To do so, you must modify his user account.

To modify the user Nick:

1. In EM Express, go to the Users page, as described in "Viewing User Accounts".
2. Select the user account NICK, and from the **Actions** menu, select **View Details**.

   The View User page appears, with the Privileges & Roles subpage displayed.
3. Click the **Quotas** subtab.

   The Quotas subpage appears.
4. Select tablespace USERS and then click **Edit**.

   The Alter Quota page appears.
5. In the **Quota** field, enter **Unlimited**.
6. Click **OK**.

   A confirmation message appears.

Locking and Unlocking User Accounts

To temporarily deny access to the database for a particular user account, you can lock the user account. If the user then attempts to connect, then the database displays an error message and does not allow the connection. You can unlock the user account when you want to permit database access again for that user. You can use Oracle Enterprise Manager Database Express (EM Express) to lock and unlock user accounts.

To lock or unlock a user account:

1. In EM Express, go to the Users page, as described in the "Viewing User Accounts" topic.
2. Click the desired user account.
3. From the **Actions** menu, select **Alter Account**.

   The Alter Account page appears.
4. Do one of the following:

   - To lock the account, enable the **Account Locked** option, and then click **OK**.
   - To unlock the account, disable the **Account Locked** option, and then click **OK**.

See Also:

Viewing User Accounts

Expiring a User Password

You can expire a user password using Oracle Enterprise Manager Database Express (EM Express). When you expirea user password, the user is prompted to change his or her password the next time that user logs in. Reasons to expire a password include the following:

- A user password becomes compromised.
- You have a security policy in place that requires users to change their passwords on a regular basis.

Note:

You can automate the automatic expiring of user passwords after a certain interval. See "Setting the Database Password Policy".

- A user has forgotten his or her password.

  In this third case, you modify the user account, assign a new temporary password, and expire the password. The user then logs in with the temporary password and is prompted to choose a new password.

To expire a user password:

1. In EM Express, go to the Users page, as described in "Viewing User Accounts".
2. Click the desired user account.
3. From the **Actions** menu, select **Alter Account**.

   The Alter Account page appears.
4. Enable **Password Expired**, and then click **OK**.

Example: Deleting a User Account

You can delete a user account using Oracle Enterprise Manager Database Express (EM Express). Suppose user Nick has moved to another department. Because it is no longer necessary for him to have access to the database, you want to delete his user account.

You must use caution when deciding to deleting a user account, because this action also deletes all schema objects owned by the user. To prevent a user from logging in to the database while keeping the schema objects intact, lock the user account instead. See "Locking and Unlocking User Accounts".

To delete user Nick:

1. In EM Express, go to the Users page, as described in "Viewing User Accounts".
2. Select the user account Nick, and then click **Drop User**. If you select the **Cascade** option, all the objects in Nick's schema will be deleted before user Nick's account is deleted.

   A confirmation page appears.
3. Click **OK** to confirm the deletion of the user account.

Setting the Database Password Policy

This section provides background information and instructions for setting the password policy for all user accounts in the database. It contains the following topics:

- About Password Policies
- Modifying the Default Password Policy

See Also:

- "Administering Database User Accounts"
- *Oracle Database 2 Day + Security Guide*

About Password Policies

When you create a user account, a default password policy is assigned to that user account. The default password policy for a newly installed database includes these directives:

- The password for the user account expires automatically in 180 days.
- The user account is locked 7 days after password expiration.
- The user account is locked for 1 day after 10 failed login attempts.

The default password policy is assigned to user accounts through a database object called a *profile*. Each user account is assigned a profile, and the profile has several attributes that describe a password policy. The database comes with a default profile (named `DEFAULT`), and unless you specify otherwise when you create a user account, the default profile is assigned to the user account.

For better database security, you may want to impose a more strict password policy. For example, you may want passwords to expire every 70 days, and you may want to lock user accounts after three failed login attempts. (A failed login attempt for a user account occurs when a user enters an incorrect password for the account.) You may also want to require that passwords be complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords. For example, you might specify that passwords must contain at least one number and one punctuation mark.

You change the password policy for every user account in the database by modifying the password-related attributes of the `DEFAULT` profile.

Note:

It is possible to have different password policies for different user accounts. You accomplish this by creating multiple profiles, setting password-related attributes differently for each profile, and assigning different profiles to different user accounts. This scenario is not addressed in this section.

See Also:

- *Oracle Database Security Guide* for an example of creating a password profile
- *Oracle Database SQL Language Reference* for more information about the SQL `CREATE PROFILE` statement

Modifying the Default Password Policy

You modify the default password policy for every database user account by modifying the password-related attributes of the profile named `DEFAULT`. You can use Oracle Enteprise Manager Database Express (EM Express) to modify the default password policy.

To modify the default password policy:

1. Log into EM Express with a user account that has privileges to manage the default password policy. An example of such a user account is `SYSTEM`.
2. In the **Security** menu, select **Profiles**.

   The Profiles page appears.

   Note:

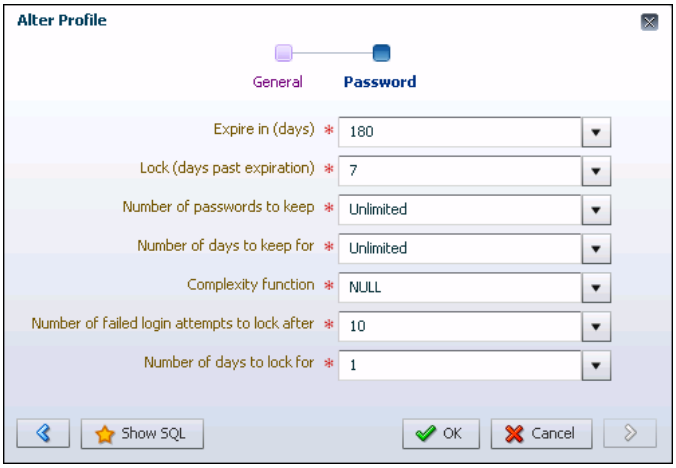
   When you are using EM Express to manage a multitenant container database (CDB) and its pluggable databases (PDBs), the Profiles option is available only at the PDB level, because profiles are at the PDB level in a CDB.

3. Select the profile named **`DEFAULT`**, and from the **Actions** menu, select **Alter Profile**.

   The Alter Profile wizard appears, with the General page showing.

4. Click the right arrow button.

   The Password page appears.



5. Change field values as required. Click the down arrow next to each field to view a list of choices. Select a value from the list, or enter a value.

6. Click **OK** to save your changes.

   A confirmation message appears.

See Also:

- "SYS and SYSTEM Users" for information about the recommended alternative to using the `SYSTEM` account for day-to-day administrative tasks
- "About Password Policies"
- *Oracle Database 2 Day + Security Guide*

Users: Oracle by Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter and includes annotated screenshots.

To view the Administering User Accounts and Security OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6287,1
```

8

Managing Schema Objects

This chapter discusses the creation and management of schema objects using SQL Developer. It contains the following sections:

About Schema Objects

A **schema** is a collection of database objects. A schema is owned by a database user and shares the same name as the user. **Schema objects** are logical structures created by users. Some objects, such as tables or indexes, hold data. Other objects, such as views or synonyms, consist of a definition only.

Note:

There is no relationship between a tablespace and a schema. Objects in the same schema can use storage in different tablespaces, and a tablespace can contain data from different schemas.

Naming Schema Objects

Every object in the database belongs to one schema and has a unique name within that schema. Multiple database objects can share the same name, if they are in different schemas. You can use the schema name to unambiguously refer to objects. For example, `hr.employees` refers to the table named `employees` in the `hr` schema. (The `employees` table is owned by `hr`.) The terms *database object* and *schema object* are used interchangeably.

When you create a database object, you must ensure that you create it in the intended schema. One method is to log in to the database as the user who owns the schema and

then create the object. Generally, you place all the objects that belong to a single application in the same schema.

A schema object name must abide by certain rules. In addition to being unique within a schema, a schema object name cannot be longer than 30 bytes and must begin with a letter. If you attempt to create an object with a name that violates any of these rules, then the database raises an error.

The DDL Tab

You can create and manipulate schema objects with SQL or with SQL Developer.

When creating schema objects using SQL Developer, you can click the **DDL** tab to display the SQL statement that is the equivalent of the schema object properties that you specified with the graphical user interface. SQL Developer submits this SQL statement to create the schema object. This option shows the statement even if it is incomplete, so you must enter all specifications for the schema object to see the complete SQL statement that SQL Developer submits.

See Also:

Oracle Database Concepts for more detailed information about schema objects

About Schema Object Management Privileges

As a database administrator (DBA), you can create, modify, and delete schema objects in your own schema and in any other schema. For purposes of this discussion, a database administrator is defined as any user who is granted the `DBA` role. This includes the `SYS` and `SYSTEM` users by default. Oracle recommends granting the `DBA` role only to those users who require administrative type access.

You can enable other users to manage schema objects without necessarily granting them DBA privileges. For example, a common scenario is to enable an application developer to create, modify, and delete schema objects in his or her own schema. To do so, you grant the `RESOURCE` role to the application developer.

See Also:

- "Example: Granting Privileges and Roles to a User Account"
- "Administering the Database with SQL-Based Management Tools"

About SQL Developer

Oracle SQL Developer is a graphical version of SQL*Plus that gives database administrators a convenient way to perform basic tasks involving schema objects. For example, you can browse, create, edit, and delete (drop) database schema objects. You can also run SQL statements and scripts, import and export table data, find invalid schema objects, and view reports.

You can connect to any target Oracle database schema using standard Oracle database authentication. Once connected, you can perform operations on schema objects.

The following topics provide basic information needed to begin using SQL Developer effectively:

- Installing and Starting SQL Developer
- Understanding the SQL Developer User Interface
- Creating a Database Connection

See Also:

You can also use SQL Developer to perform other database tests, such as unit testing, migrations, and data modeling.

See *Oracle SQL Developer User's Guide* for more information on other SQL Developer features.

Installing and Starting SQL Developer

To install and start SQL Developer, you download a ZIP file and unzip it into a desired parent directory or folder, and then type a command or double-click a file name.

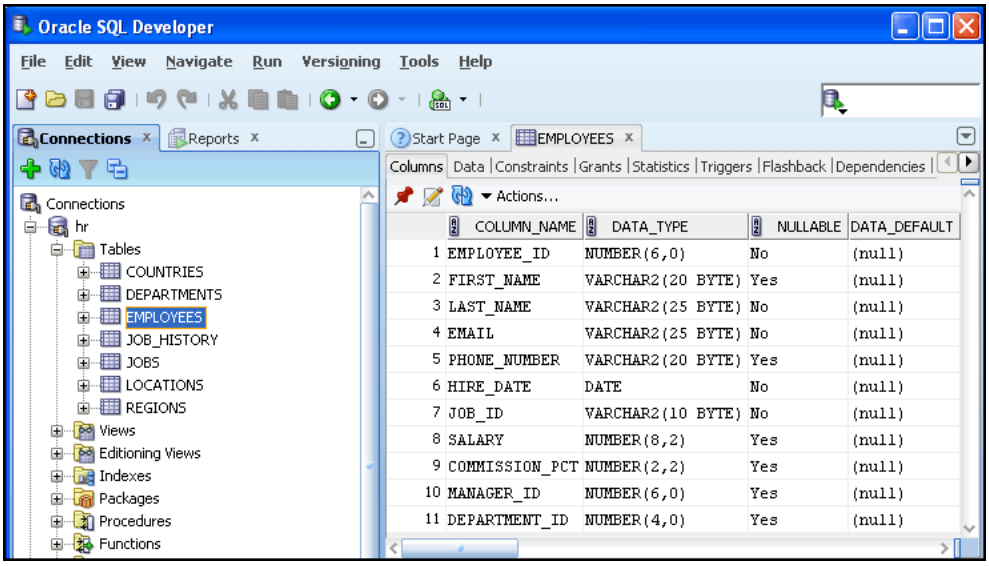
See Also:

Oracle SQL Developer User's Guide for detailed instructions on installing and starting SQL Developer.

Understanding the SQL Developer User Interface

When you start SQL Developer, the SQL Developer window appears.

The SQL Developer window generally uses the left side for navigation to find and select objects, and the right side to display information about selected objects.



The menus along the top of the page contain standard entries, plus entries for features specific to SQL Developer.

The left side of the SQL Developer window has tabs and panes for the Connections and Reports navigators, icons for performing actions, and a hierarchical tree display for the currently selected navigator.

In the figure, the HR database connection appears in the Connections navigator, and the schema objects for the HR schema appear in the metadata tree.

The metadata tree in the Connections navigator displays all the objects (categorized by object type) accessible to the defined connections. To select an object, expand the appropriate tree node or nodes, then click the object.

The right side of the SQL Developer window has tabs and panes for objects that you select or open. For example, the object pane in the figure displays information about a table named EMPLOYEES. (If you hold the mouse pointer over the tab label -- EMPLOYEES in this figure -- a tooltip displays the object's owner and the database connection.)

See Also:

Oracle SQL Developer User's Guide for more information about the SQL Developer user interface.

Creating a Database Connection Using SQL Developer

A database connection is a SQL Developer object that specifies the necessary information for connecting to a specific database as a specific user of that database. You must have at least one database connection (existing, created, or imported) to use SQL Developer.

To create a database connection:

1. In the Connections navigator in SQL Developer, right-click the Connections node and select **New Connection**.

   The New / Select Database Connection dialog box appears, with the Oracle tab displayed.

2. Enter the following information:

   - In the **Connection Name** field, enter the name to use for this database connection.
   - In the **Username** field, enter the name of the user for whom this database connection is being created.
   - In the **Password** field, enter the password for the user.
   - In the **Connection Type** field, select the database connection type.

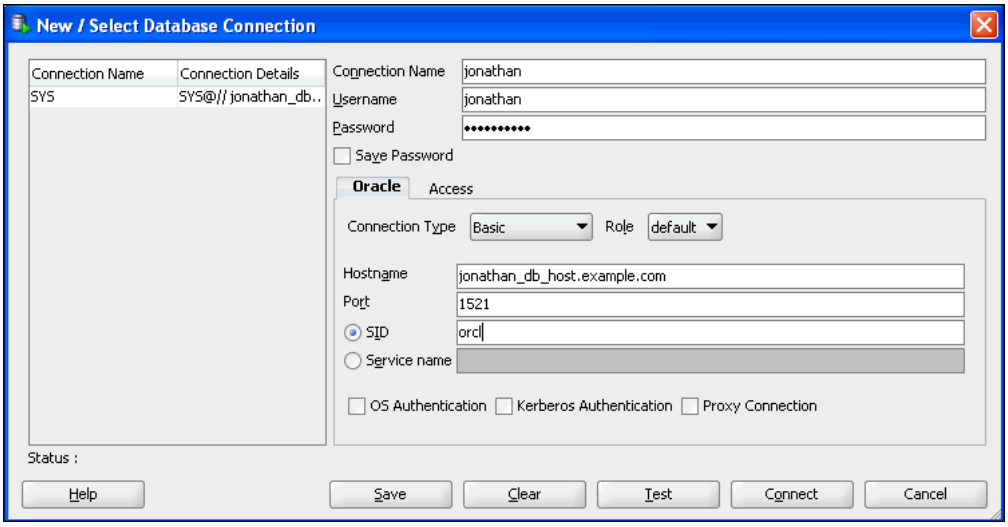
     The connection types are:

     - Basic
     - TNS
     - LDAP
     - Advanced

- Local/Bequeath

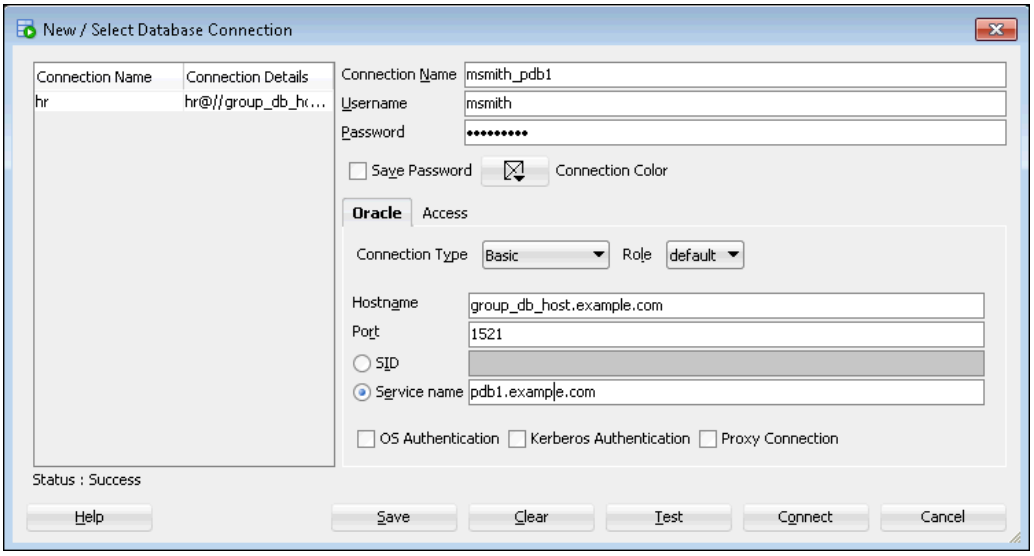
When you choose a connection type, the fields below will change to be appropriate for the selected connection type. This example describes the fields for the Basic connection type.

- In the **Role** field, select `Default` or `SYSDBA`, based on the role assigned to the user.
- In the **Hostname** field, enter the name of the host where the database is located.
- In the **Port** field, enter the port for the database.
- In the **SID** field, enter the SID for the database (when the database connection is for a non-CDB user or for a multitenant container database (CDB) user):



When a database connection to a non-CDB or CDB is created for an administrative user such as SYS, SYSDBA is typically specified in the **Role** field for the connection.

- In the **Service name** field, enter the service name for the pluggable database (PDB), including the domain name (when the database connection is for a PDB user):



When a database connection to a PDB is created for an administrative user such as SYS, SYSDBA is typically specified in the **Role** field for the connection.

3. Optionally, click **Test** to test that the data you provided will allow the specified user to connect to the database.
4. When you are finished, click **Connect** to connect using the database connection, or click **Save** to save the database creation.

See Also:

- *Oracle SQL Developer User's Guide* for more information about creating, editing, exporting, and importing database connections.
- *Oracle Database Concepts* for an overview of the multitenant architecture introduced in Oracle Database 12*c*, which enables an Oracle database to function as a multitenant container database (CDB) that includes zero, one, or many customer created pluggable databases (PDBs). A non-CDB is a traditional Oracle database that cannot contain PDBs. All Oracle databases before Oracle Database 12*c* were non-CDBs.
- *Oracle Database Administrator's Guide* for complete information about creating and configuring a CDB.

Managing Tables

The following topics discuss database tables and how to create and modify them using SQL Developer:

- About Tables
- Viewing Tables
- Viewing Table Data
- Example: Creating a Table

About Tables

The table is the basic unit of data storage in an Oracle database. It holds all user-accessible data. Each table is made up of columns and rows. In the `employees` table, for example, there are columns called `last_name` and `employee_id`. Each row in the table represents a different employee, and contains a value for `last_name` and `employee_id`.

When you create a table, you specify the table type, and define its columns and constraints. **Constraints** are rules that help preserve data integrity.

This section contains the following topics:

About Table Types

The most common type of table in an Oracle database is a *relational* table, which is structured with simple columns similar to the `employees` table. Two other table types are supported: *object* tables and *`XMLType`* tables. Any of the three table types can be defined as *permanent* or *temporary*. Temporary tables hold session-private data that exists only for the duration of a transaction or session. They are useful in applications where a results set must be held temporarily in memory, perhaps because the results set is constructed by running multiple operations.

You can build relational tables in either *heap* or *index-organized* structures. In heap structures, the rows are not stored in any particular order. In index-organized tables, the row order is determined by the values in one or more selected columns. For some applications, index-organized tables provide enhanced performance and more efficient use of disk space.

This section describes permanent, heap-organized tables. For information about other table types and when to use them, see *Oracle Database Administrator's Guide* and *Oracle Database Concepts*. For the syntax required to create tables with SQL, see *Oracle Database SQL Language Reference*.

About Table Column Attributes

You define table columns to hold your data. When you create a column, you specify the following attributes:

Data Type

The data type attribute defines the kind of data to be stored in the column. When you create a table, you must specify a data type for each of its columns.

Data types define the domain of values that each column can contain. For example, `DATE` columns cannot accept the value `February 29` (except for a leap year) or the values `2` or `SHOE`. Each value subsequently inserted in a column assumes the column data type. For example, if you insert `17-JAN-2004` into a date column, then Oracle Database treats that character string as a date value after verifying that it converts to a valid date.

Table 1 lists some common Oracle Database built-in data types.

Table 1 Common Data Types

Data Type	Description
`VARCHAR2(`*`size`* `[BYTE|CHAR])`	Variable-length character string having a maximum length of *`size`* bytes or characters. A column to hold postal codes for different countries, for example, might be restricted to 12 bytes by defining it as `VARCHAR2(12)`. You can use the `CHAR` qualifier, for example `VARCHAR2(10 CHAR)`, to indicate the maximum length in characters, without regard for the number of bytes required. This is especially useful for languages that use characters with double-byte and triple-byte lengths. The `BYTE` and `CHAR` qualifiers override the setting of the `NLS_LENGTH_SEMANTICS` parameter, which has a default of bytes. The maximum size is 4000 bytes or characters. The minimum is 1 byte or 1 character. You must specify *`size`* for `VARCHAR2`. See *Oracle Database Globalization Support Guide* for more information.
`NUMBER (`*`p`*`,`*`s`*`)`	Number having precision *`p`* and scale *`s`*. Precision sets the maximum number of digits in the number, and scale defines how many of the digits are to the right of the decimal point. For example, a field to hold monetary values might be defined as `NUMBER(12,2)`, providing 10 digits for the primary unit of currency (dollars, pounds, marks, and so on) and two digits for the secondary unit (cents, pennies, pfennigs, and so on). The precision *`p`* can range from 1 to 38. The scale *`s`* can range from -84 to 127.
`DATE`	A composite value that includes both a date and time component. For each `DATE` value, the database stores the following information: century, year, month, day, hour, minute, and second. When entering a date into a table column of type `DATE`, you must use the format specified by the `NLS_DATE_FORMAT` initialization parameter. The `NLS_TERRITORY` initialization parameter determines the default value of the `NLS_DATE_FORMAT` parameter. For example, in the United States, the `NLS_DATE_FORMAT` parameter defaults to '`DD-MON-RR`'. You must therefore enter a date in the format '`11-JAN-06`'. Because this format does not include a time component, the time defaults to 12:00:00 a.m. (midnight). You can also use the `TO_DATE` function, which converts a character string to a date, to include a time component or to enter a date in another format. The valid date range is from January 1, 4712 BC to December 31, 9999 AD.

Data Type	Description
CLOB	A character large object (CLOB) containing single-byte or multibyte characters. Both fixed-width and variable-width character sets are supported, both using the database character set. The maximum size is (4 gigabytes - 1) * (database block size). For example, for a block size of 32K, the maximum CLOB size is 128 terabytes.

NOT NULL Column Constraint

Constraints determine valid values for the column. In SQL Developer, the only constraint you can define at the column level in the Create Table dialog box page is the NOT NULL constraint, which requires that a value be included in the column whenever a row is inserted or updated. Unlike other constraints described in "About Table-Level Constraints", which can be defined as part of the column definition or part of the table definition, the NOT NULL constraint must be defined as part of the column definition.

Use a NOT NULL constraint when data must be supplied for a column for the integrity of the database. For example, if all employees must belong to a specific department, then the column that contains the department identifier must be defined with a NOT NULL constraint. However, do not define a column as NOT NULL if the data can be unknown or may not exist when rows are added or changed. An example of a column for which you must not use a NOT NULL constraint is the second, optional line in a mailing address.

The database automatically adds a NOT NULL constraint to the column or columns included in the primary key of a table.

Default Value

This value is automatically stored in the column whenever a new row is inserted without a value being provided for the column. You can specify a default value as a literal or as an expression. However, there are limitations on how you construct the expression. See *Oracle Database SQL Language Reference* for details.

Encryption

You can enable automatic encryption for column data. See the discussion of Transparent Data Encryption in *Oracle Database 2 Day + Security Guide* for more information.

About Table-Level Constraints

In an Oracle database, you can apply rules to preserve the integrity of your data. For example, in a table that contains employee data, the employee name column cannot accept NULL as a value. Similarly, in this table, you cannot have two employees with the same ID.

Oracle Database enables you to apply data integrity rules called **constraints**, both at the table level and at the column level. Any SQL statement that attempts to insert or update a row that violates a constraint results in an error, and the statement is rolled back. Likewise, any attempt to apply a new constraint to a populated table also results in an error if any existing rows violate the new constraint.

The types of constraints that you can apply at the table level are as follows:

- **Primary Key**—Requires that a column (or combination of columns) be the unique identifier of the row. A primary key column does not allow NULL values.

- **Unique Key**—Requires that no two rows can have duplicate values in a specified column or combination of columns. The set of columns is considered to be a unique key.
- **Check**—Requires that a column (or combination of columns) satisfy a condition for every row in the table. A check constraint must be a Boolean expression. It is evaluated each time that a row is inserted or updated. An example of a check constraint is: `SALARY > 0`.
- **Foreign Key**—Requires that for a particular column (or combination of columns), all column values in the child table exist in the parent table. The table that includes the foreign key is called the dependent or **child** table. The table that is referenced by the foreign key is called the **parent** table. An example of a foreign key constraint is where the department column of the employees table must contain a department ID that exists in the parent department table.

Constraints can be created and usually modified with different statuses. The options include enabled or disabled, which determine if the constraint is checked when rows are added or modified, and deferred or immediate, which cause constraint validation to occur at the end of a transaction or at the end of a statement, respectively.

See Also:

- *Oracle Database Concepts* for more information about constraints

Other Table Creation Considerations

This section describes some additional considerations for creating tables. It contains the following topics:

- User-Defined Types and Large Objects (LOBs)
- Partitioned Tables and Indexes
- Physical Storage Attributes
- Compressed Tables

User-Defined Types and Large Objects (LOBs)

Your new table can include one or more columns defined with user-defined types. **User-defined types** enable a single column in a single row to contain multiple values. The multiple values can be represented as arrays, nested tables, or objects, where an object type represents a real-world entity such as a purchase order. (Retrieving a purchase order–type column value could return a *record* that contains purchase order number, customer number, amount, and so on.) User-defined types are created with the `CREATE TYPE` statement and are described in detail in *Oracle Database SQL Language Reference*.

Large object (LOB) columns are used to contain unstructured data (such as text or streaming video), and can hold terabytes of information.

To create a LOB column using SQL Developer, click the **Advanced** checkbox when creating a table. Then click **LOB Parameters** to see the options available when creating a LOB column. For details about creating LOB columns, see *Oracle Database SQL Language Reference*.

Partitioned Tables and Indexes

You can *partition* tables and indexes. Partitioning helps to support very large tables and indexes by enabling you to divide the tables and indexes into smaller and more manageable pieces called **partitions**. SQL queries and DML statements do not have to be modified to access partitioned tables and indexes. Partitioning is transparent to the application.

After partitions are defined, certain operations become more efficient. For example, for some queries, the database can generate query results by accessing only a subset of partitions, rather than the entire table. This technique (called **partition pruning**) can provide order-of-magnitude gains in improved performance. In addition, data management operations can take place at the partition level, rather than on the entire table. This results in reduced times for operations such as data loads; index creation and rebuilding; and backup and recovery.

Each partition can be stored in its own tablespace, independent of other partitions. Because different tablespaces can be on different disks, this provides a table structure that can be better tuned for availability and performance. Storing partitions in different tablespaces on separate disks can also optimize available storage usage, because frequently accessed data can be placed on high-performance disks, and infrequently retrieved data can be placed on less expensive storage.

Partitioning is useful for many types of applications that manage large volumes of data. Online transaction processing (OLTP) systems often benefit from improvements in manageability and availability, while data warehousing systems benefit from increased performance and manageability.

To specify partitioning options using SQL Developer, click the **Advanced** checkbox when creating a table. Then click **Partitioning** to see the partitioning options available. For details about partitioning, see *Oracle Database SQL Language Reference*.

Physical Storage Attributes

You can specify several storage attributes for a table. For example, you can specify the initial size of the table on disk. For more information about setting storage attributes for a table, see *Oracle Database Administrator's Guide* and *Oracle Database SQL Language Reference*.

To specify storage attributes for a table using SQL Developer, click the **Advanced** checkbox when creating a table, then click **Table Properties**, and then click **Storage Options**.

Compressed Tables

Table Compression is suitable for both OLTP applications and data warehousing applications. Compressed tables require less disk storage and result in improved query performance due to reduced I/O and buffer cache requirements. Compression is transparent to applications and incurs minimal overhead during bulk loading or regular DML operations such as INSERT, UPDATE or DELETE.

To configure table compression using SQL Developer, click the **Advanced** checkbox when creating a table. Then click **Table Properties** and enable the **Compression** option.

See Also:

- *Oracle Database Administrator's Guide* for design and management considerations for different table types
- *Oracle Database Concepts* and *Oracle Database VLDB and Partitioning Guide* for more information about partitioned tables and indexes
- *Oracle Database SecureFiles and Large Objects Developer's Guide* for more information about SecureFiles LOBs and BasicFiles LOBs
- "Example: Creating a Table"

Viewing Tables

You can use SQL Developer to list all the tables in a specified schema, and to view the definitions of individual tables.

To view tables:

1. In the Connections navigator in SQL Developer, navigate to the Tables node for the schema that includes the table you want to display.

   If the view is in your own schema, navigate to the Tables node in your schema.

   If the table you want to display is in another user's schema, navigate to the Other Users node, expand it, find the name of the schema the table is in, and navigate to the Tables node.

   Examples of schema names include SYS and HR.

   Note:

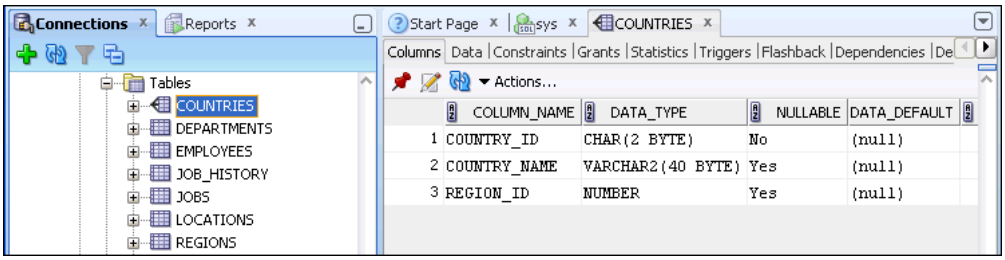
   You must have the necessary privileges to view other schemas and the objects in those schemas.

2. Open the Tables node.

   The list of tables in the schema appears.

3. Click the name of the table that you want to display.

   A tab with the table name appears in the object pane, with the Columns subtab displayed. You can view the table definition on this tab.



See Also:

" About Tables"

Viewing Table Data

Besides viewing table names and table definitions in SQL Developer, you can view the data stored in the table, and the SQL statement used to display the data. You can also change the SQL statement to alter the results set.

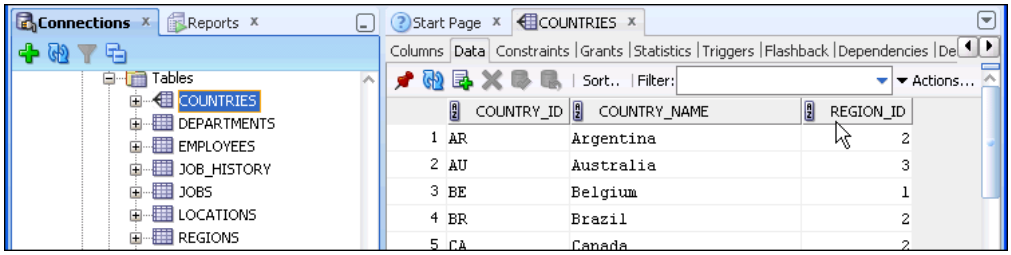
To view table data:

1. In SQL Developer, search for a table as described in "Viewing Tables". For example, search for the tables in the HR schema.
2. Select the table that contains the data.

   For example, select `countries`.

   A tab with the table name appears in the object pane, with the Columns subtab displayed.
3. In the object pane, click the Data subtab.

   The data for the table appears on the Data subtab.



4. (Optional) Click a column name to sort the data by that column.
5. (Optional) Click the SQL subtab to view the SQL statement that defines the table.

You can also write and submit your own SQL `SELECT` statement to see the contents of a table. You can run SQL statements by starting a SQL Worksheet session in SQL Developer. To do so, from the **Tools** menu, select **SQL Worksheet**.

A detailed description of the `SELECT` statement is in *Oracle Database SQL Language Reference*.

See Also:

" About Tables"

Example: Creating a Table

You can use SQL Developer to create a table.

In the following example, you create a table called `purchase_orders` in the `HR` schema. The table has the following columns:

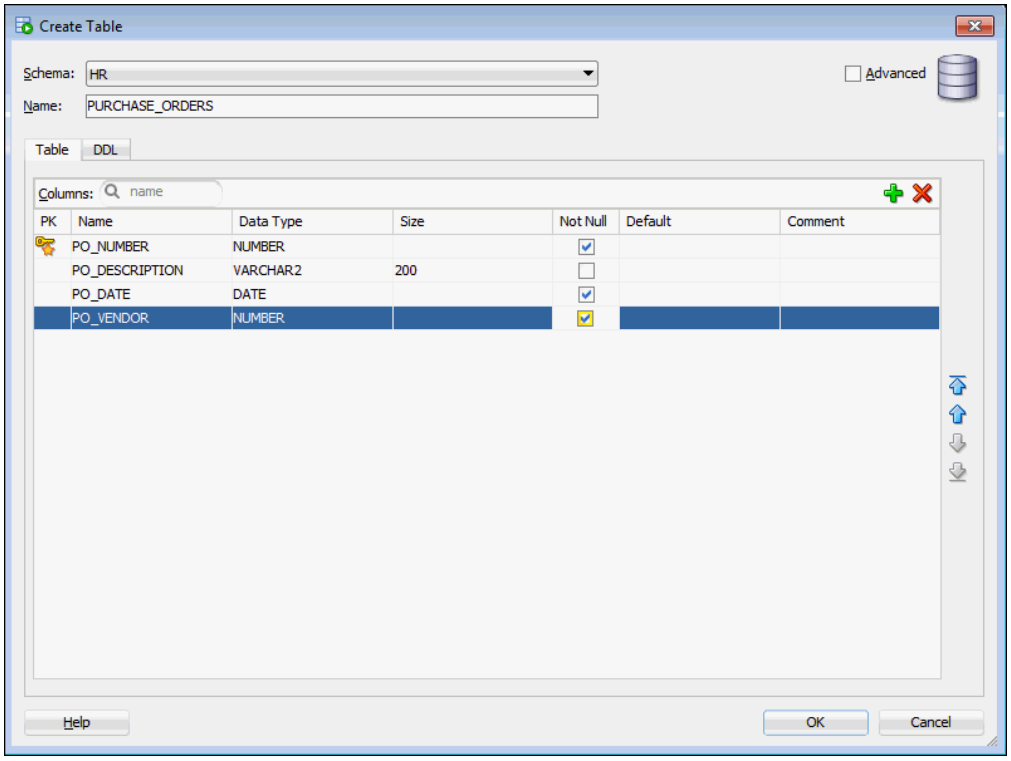
Column Name	Data Type	Size	Not Null	Primary Key
PO_NUMBER	NUMBER		Yes	Yes
PO_DESCRIPTION	VARCHAR2	200	No	
PO_DATE	DATE		Yes	
PO_VENDOR	NUMBER		Yes	

To create the PURCHASE_ORDERS table in the HR schema:

1. In SQL Developer, navigate to the Tables node in the schema where you want to create the new table, as described in "Viewing Tables".
2. Right-click the Tables node and select **New Table**.

   The Create Table dialog box appears, with the Table tab displayed.

   You can create simple tables quickly using only the default options on the Table tab. If you want to create more advanced tables, select the **Advanced** checkbox, which provides many more options for table creation. Unselect the **Advanced** checkbox to return to the default options on the Table tab.
3. In the **Schema** field, select HR.
4. In the **Name** field, enter PURCHASE_ORDERS.
5. In the **Columns** section, enter column information for each of the columns in the PURCHASE_ORDERS table as specified in the table in the introduction to this topic. For example, for the first column in the PURCHASE_ORDERS table, enter the name PO_NUMBER and the data type NUMBER, select the **Not Null** check box, and click in the **PK** column to indicate that PO_NUMBER is the primary key for the table. Then click the green plus sign icon to add information for the next column. Continue in this manner until you have added the information for all of the columns to PURCHASE_ORDERS.



6. Click **OK**.

   The PURCHASE_ORDERS table appears under the Tables node in the Connections navigator.

See Also:

" About Tables"

Modifying Table Attributes

You can use SQL Developer to add and delete table columns and to manage table constraints. This section contains the following topics:

- Example: Adding Table Columns
- Example: Deleting a Table Column
- Example: Adding a New Table Constraint
- Example: Modifying an Existing Table Constraint
- Example: Deleting a Table Constraint

See Also:

" About Tables"

Example: Adding Table Columns

In this example, you use SQL Developer to add columns to the `purchase_orders` table that you created previously in "Example: Creating a Table". The two new columns are named `po_date_received` and `po_requestor_name`.

To add columns to the PURCHASE_ORDERS table:

1. In SQL Developer, navigate to the Tables node in the HR schema, following the instructions in "Viewing Tables".
2. Expand the Tables node.

   The list of tables in the schema appears.
3. Right-click the PURCHASE_ORDERS table and select **Edit**.

   The Edit Table dialog box appears.
4. At the top right of the Columns section, click the green plus sign icon.

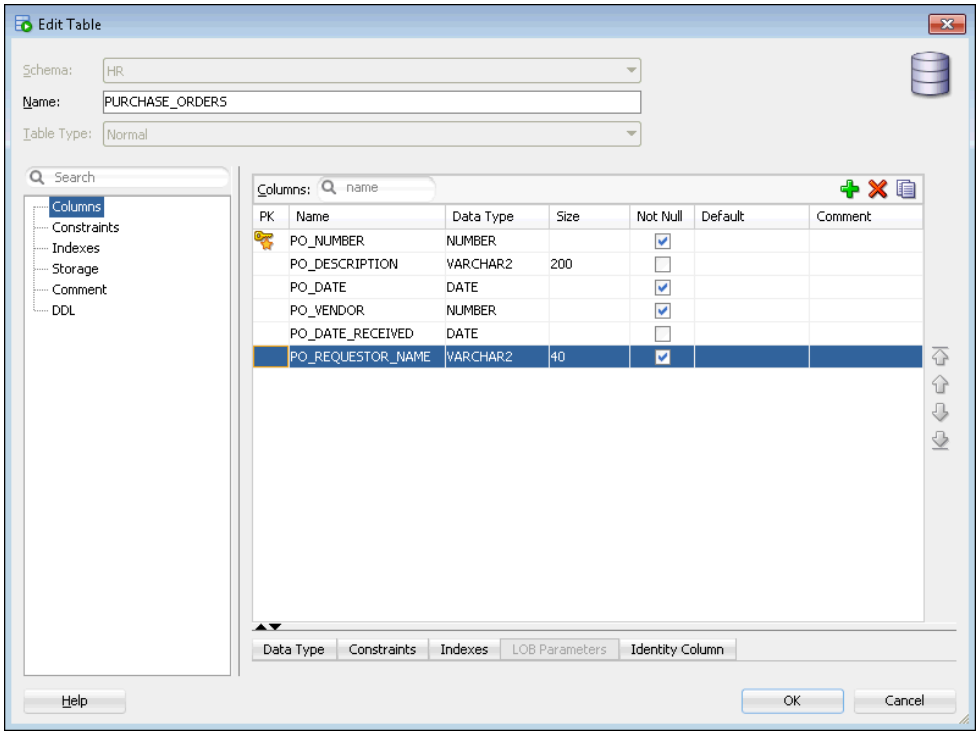
   A new blank column appears at the bottom of the list of columns.
5. In the Columns section, enter the following information about the new `po_date_received` column:

Field Name	Value
Name	PO_DATE_RECEIVED
Type	DATE

6. At the top right of the Columns section, click the green plus sign icon again.

   A new blank column appears at the bottom of the list of columns.
7. In the Column Properties section, enter the following information about the new `po_requestor_name` column:

Field Name	Value
Name	PO_REQUESTOR_NAME
Type	VARCHAR2
Size	40



8. Click **OK**.

   The PURCHASE_ORDERS table appears under the Tables node in the Connections navigator.

9. To view the new columns, click PURCHASE_ORDERS in the Connections navigator. Then, on the PURCHASE_ORDERS tab in the objects pane, click the Columns subtab to view the new columns.

See Also:

" About Tables"

Example: Deleting a Table Column

In this example, you use SQL Developer to delete the `po_requestor_name` column that you added to the `purchase_orders` table in "Example: Adding Table Columns".

To delete the PO_REQUESTOR_NAME column:

1. In SQL Developer, navigate to the PURCHASE_ORDERS table in the HR schema, following the instructions in "Viewing Tables".

2. Right-click the PURCHASE_ORDERS table and select **Edit**.

   The Edit Table dialog box appears.

3. In the Columns section, click the PO_REQUESTOR_NAME column, and then click the red X icon.

   The PO_REQUESTOR_NAME column is removed from the list of columns.

4. Click **OK**.
5. On the PURCHASE_ORDERS table tab in the object pane, click the Columns subtab to view the list of columns in the table.

See Also:

" About Tables"

Example: Adding a New Table Constraint

In this example, you use SQL Developer to add a table constraint to the `purchase_orders` table that you created in "Example: Creating a Table". To enforce the rule that the `po_date_received` value must be either the same day as, or later than, the value of `po_date`, you add a check constraint.

Note:

You can also add constraints during table creation, as shown in "Example: Creating a Table". In that example, you added a primary key constraint.

To add a table constraint to the PURCHASE_ORDERS table:

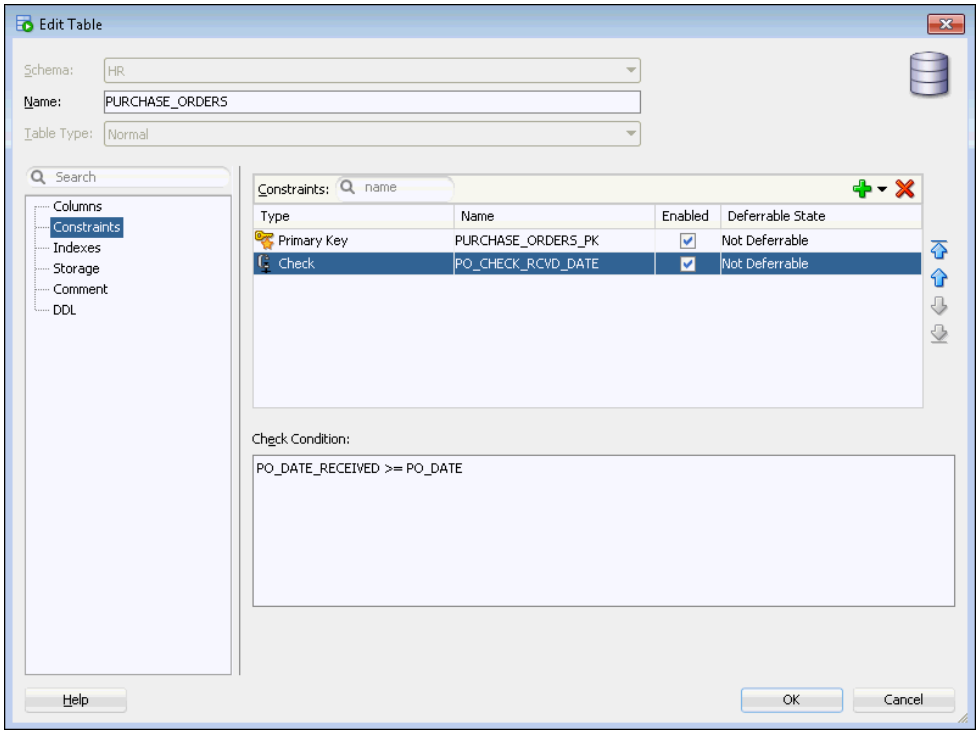
1. Navigate to the PURCHASE_ORDERS table in the HR schema, following the instructions in "Viewing Tables".
2. Right-click the PURCHASE_ORDERS table and select **Edit**.

   The Edit Table dialog box appears.
3. Under the Search box, click **Constraints**.
4. To the right of the Constraints section, click the green plus sign icon and choose **New Check Constraint**.

   In the Constraints section, a new row appears at the bottom of the list of constraints.
5. In the **Name** field for the new constraint, enter PO_CHECK_RCVD_DATE.
6. Leave the check mark in the **Enabled** column and leave **Not Deferrable**.
7. In the **Check Condition** section, enter the following condition for this constraint:

   ```
   PO_DATE_RECEIVED >= PO_DATE
   ```

   This expression indicates that PO_DATE_RECEIVED must be greater than or equal to PO_DATE. For date columns, this is equivalent to stating that PO_DATE_RECEIVED must be on the same day as, or later than, PO_DATE.



8. Click **OK**.
9. On the PURCHASE_ORDERS tab in the object pane, click the Constraints subtab to view the current constraints in the table.

See Also:

- " About Tables"
- "About Table-Level Constraints"

Example: Modifying an Existing Table Constraint

There are a few ways in which you can modify a table constraint. You can change the status of an existing table constraint, for example, from an enabled state to a disabled state. In this example, you use SQL Developer to disable the check constraint that you created for the `purchase_orders` table in "Example: Adding a New Table Constraint".

To disable a constraint for the PURCHASE_ORDERS table:

1. In SQL Developer, navigate to the `PURCHASE_ORDERS` table in the HR schema, following the instructions in "Viewing Tables".
2. Right-click the `PURCHASE_ORDERS` table and select **Constraint**, and then **Disable Single**.

   The Disable Single dialog box appears.
3. In the **Constraint** field, select `PO_CHECK_RCVD_DATE`.

4. Click **Apply**.

   A confirmation message appears advising that the constraint has been disabled.

5. On the PURCHASE_ORDERS tab in the object pane, click the Constraints subtab to view the PO_CHECK_RCVD_DATE constraint. The Status column for this constraint shows a value of DISABLED.

See Also:

- " About Tables"
- "About Table-Level Constraints"

Example: Deleting a Table Constraint

You can delete constraints from a table with SQL Developer. Deleting a table constraint may cause the deletion of other constraints. For example, if you delete the primary key constraint from a table (the parent table) that is referenced in a foreign key constraint in another table (the child table), then the foreign key constraint in the child table is also deleted through a cascading delete mechanism.

In this example, you delete the check constraint that you created for the `purchase_orders` table in "Example: Adding a New Table Constraint".

To delete a constraint from the PURCHASE_ORDERS table:

1. In SQL Developer, navigate to the `PURCHASE_ORDERS` table in the HR schema, following the instructions in "Viewing Tables".
2. Right-click the `PURCHASE_ORDERS` table and select **Constraint**, and then **Drop**.

   The Drop dialog box appears.

3. In the **Constraint** field, select `PO_CHECK_RCVD_DATE`.
4. Click **Apply**.

   A confirmation message appears advising that the constraint has been dropped.

5. On the PURCHASE_ORDERS tab in the object pane, click the Constraints subtab. The PO_CHECK_RCVD_DATE constraint no longer appears in this table.

See Also:

- *Oracle Database Concepts* for more information about the cascading delete mechanism
- " About Tables"
- "About Table-Level Constraints"

Example: Loading Data into a Table

You can use SQL Developer to load data into a table. You can load data from an .xls file or a .csv file into the table.

In this example, you load data into the PURCHASE_ORDERS table that you created in "Example: Creating a Table". For simplicity, this example loads only three rows.

To prepare for this example, you must create a text file named `load.csv` on the file system of the database host computer or on the file system of your local computer. The contents of the file should be as follows:

```
1,Office Equipment,25-MAY-2012,1201,13-JUN-2012
2,Computer System,18-JUN-2012,1201,27-JUN-2012
3,Travel Expense,26-JUN-2012,1340,11-JUL-2012
```

Note:

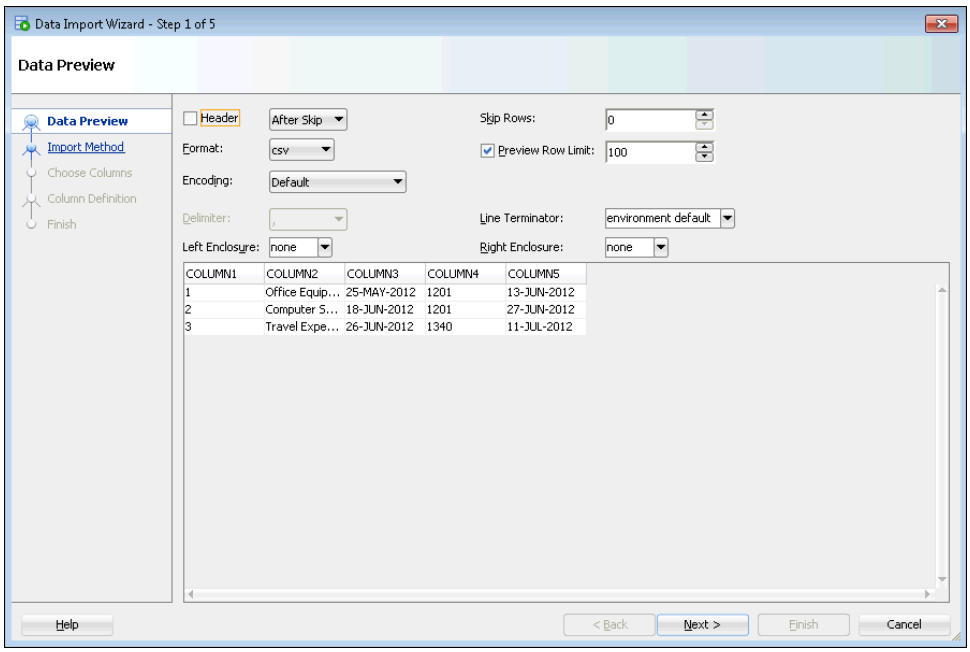
This example assumes that the columns in the PURCHASE_ORDERS table are the following: PO_NUMBER, PO_DESCRIPTION, PO_DATE, PO_VENDOR, and PO_DATE_RECEIVED. If your PURCHASE_ORDERS table does not have all these columns (or has additional columns), then modify the data in the text file accordingly.

To load data into the PURCHASE_ORDERS table:

1. In SQL Developer, navigate to the PURCHASE_ORDERS table in the HR schema, following the instructions in "Viewing Tables".
2. Right-click the PURCHASE_ORDERS table and select **Import Data**.

   The Open dialog box appears.
3. Navigate to and select the `load.csv` file that includes the data you want to import into the table, and then click **Open**.

   The Data Import Wizard appears, with the Data Preview page displayed.
4. Make sure that **Header** is deselected, **Format** is set to `csv`, **Line Terminator** is set to `environment default`, and that **Left Enclosure** and **Right Enclosure** are set to `none`. Then click **Next**.



The Import Method page appears.

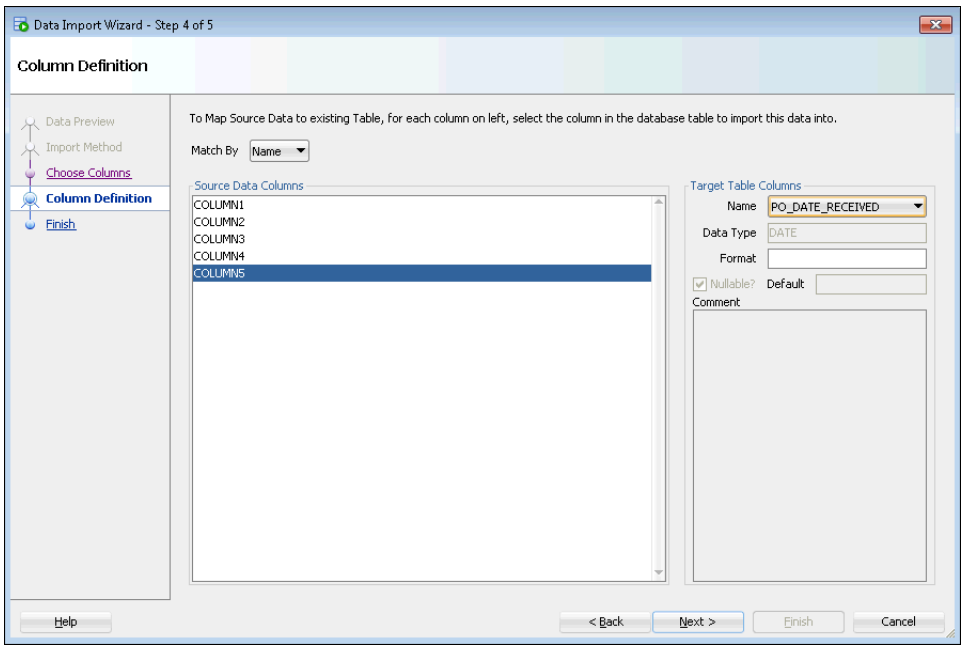
5. On this page, select:
   - `Insert` in the **Import Method** field.
   - `PURCHASE_ORDERS` in the **Table Name** field.
   - A value in the **Import Row Limit** field that is greater than the number of rows in your `.csv` file.
6. Click **Next**.

   The Choose Columns page appears.
7. Move the columns that you want to import into the **Selected Columns** list, and arrange them in the order you want.
8. Click **Next**.

   The Column Definition page appears.
9. Map the source data from the `.csv` file to the existing table. For each source data column on the left, select the column on the right to import this data into.

   For example, in the **Source Data Columns** table on the left, select the first column, and then in the **Target Table Columns** table on the right, select in the **Name** field the name of the column in the database table that will store that data. Map the data for each of the columns in the **Source Data Columns** table to the appropriate column in the **Target Table Columns** table.

   The figure shows the data in the last column in the **Source Table Columns** table being mapped to the last column (`PO_DATE_RECEIVED`) in the **Target Table Columns** table. At this point, the previous four columns in the **Source Table Columns** table have already been mapped to the appropriate columns in the **Target Table Columns** table.



10. Click **Next**.

    The Finish page appears.

11. Click **Finish**.

    The data is imported into the table.

See Also:

" About Tables"

Deleting a Table

If you no longer need a table, then you can delete it using SQL Developer. When you delete a table, the database deletes the data and dependent objects of the table (such as indexes), and removes the table from the data dictionary.

When you delete a table from a locally managed tablespace that is not the `SYSTEM` tablespace, the database does not immediately reclaim the space associated with the table. Instead, it places the table and any dependent objects in the *recycle bin*. You can then restore the table, its data, and its dependent objects from the recycle bin if necessary. You can view the contents of the recycle bin by clicking **Recycle Bin** on the Tables page. Note that users can see only tables that they own in the recycle bin. See *Oracle Database Administrator's Guide* for more information about the recycle bin, including how to view, purge, and recover tables for which you are not the owner.

To delete a table:

1. In SQL Developer, navigate to the `PURCHASE_ORDERS` table in the HR schema, following the instructions in "Viewing Tables".
2. Right-click the `PURCHASE_ORDERS` table and select **Table** and then **Drop**.

   The Drop dialog box appears.
3. Select **Cascade Constraints** and **Purge**.
4. Click **Apply**.

   A confirmation message appears.

See Also:

" About Tables"

Managing Indexes

The following topics describe how to create and manage indexes using SQL Developer:

- About Indexes
- Viewing Indexes

- Example: Creating an Index
- Example: Deleting an Index

About Indexes

Indexes are optional schema objects that are associated with tables. You create indexes on tables to improve query performance. Just as the index in a guide helps you to quickly locate specific information, an Oracle Database index provides quick access to table data.

You can create as many indexes on a table as you need. You create each index on one or more columns of a table. For example, in a purchase orders table, if you create an index on the vendor number column, then you can sequentially access the rows of the table in vendor number order, without having to actually sort the rows. Additionally, you can directly access all purchase orders issued to a particular vendor without having to scan the entire table.

After an index is created, it is automatically maintained and used by the database. Changes to the data or structure of a table, such as adding new rows, updating rows, or deleting rows, are automatically incorporated into all relevant indexes. This is transparent to the user.

Some indexes are created implicitly through constraints that are placed on a table. For example, the database automatically creates an index on the columns of a primary key constraint or unique key constraint.

The following topics provide more background information about indexes:

- Indexes and Performance
- Index Attributes

See Also:

- "Viewing Indexes"
- "Example: Creating an Index"
- *Oracle Database Concepts*

Indexes and Performance

Indexes generally improve the performance of queries and DML statements that operate on a single, existing row or a small number of existing rows. However, too many indexes can increase the processing overhead for statements that add, modify, or delete rows.

Before you add additional indexes, examine the performance of your database for queries and DML. You can then compare performance after the new indexes are added.

See Also:

Oracle Database Testing Guide for information about using the SQL Performance Analyzer to analyze the SQL performance impact of any type of schema or system changes

Index Attributes

Indexes can be created in several ways, using various combinations of index attributes. The primary index attributes are the following:

Standard (B-tree) and Bitmap

A standard, B-tree index contains an entry for each value in the index key along with a disk address of the row where the value is stored. A B-tree index is the default and most common type of index in an Oracle database.

A bitmap index uses strings of bits to encapsulate values and potential row addresses. It is more compact than a B-tree index and can perform some types of retrieval more efficiently. For general use, however, a bitmap index requires more overhead during row operations on the table and should be used primarily for data warehouse environments, as described in *Oracle Database Data Warehousing Guide*.

Ascending and Descending

The default search through an index is from lowest to highest value, where character data is sorted by ASCII values, numeric data from smallest to largest number, and date from the earliest to the latest value. This default search method is performed in indexes created as ascending indexes. You can cause index searches to reverse the search order by creating the related index with the descending option.

Column and Functional

Typically, an index entry is based on the value or values found in the column or columns of a table. This is a column index. Alternatively, you can create a function-based index in which the indexed value is derived from the table data. For example, to find character data that can be in various combinations of upper and lowercase letters, you can use a function-based index based on the `UPPER()` function to look for the values as if they were all in uppercase characters.

Single-Column and Concatenated

You can create an index on just one column, which is called a **single-column index**, or on multiple columns, which is called a **concatenated index**. Concatenated indexes are useful when all the index columns are likely to be included in the `WHERE` clause of frequently executed SQL statements.

Nonpartitioned and Partitioned

As with tables, you can partition an index. In most situations, it is useful to partition an index when the associated table is partitioned, and to partition the index using the same partitioning scheme as the table. (For example, if the table is range-partitioned by sales date, then you create an index on sales date and partition the index using the same ranges as the table partitions.) This is known as a **local** partitioned index. However, you do not have to partition an index using the same partitioning scheme as its table. You can also create a nonpartitioned, or **global**, index on a partitioned table.

See Also:

- *Oracle Database Concepts* for design and management considerations of different index types
- *Oracle Database SQL Language Reference* for the syntax to create indexes
- *Oracle Database VLDB and Partitioning Guide* for more information about partitioned tables and indexes

Viewing Indexes

You use SQL Developer to view the indexes in your database.

To view indexes:

1. In the Connections navigator in SQL Developer, navigate to the Indexes node for the schema that includes the index you want to view.

   If the index is in your own schema, navigate to the Indexes node in your schema.

   If the index you want to view is in another user's schema, navigate to the Other Users node, expand it, find the name of the schema the index is in, and navigate to the Indexes node.

   Examples of schema names include `SYS` and `HR`.

   Note:

   You must have the necessary privileges to view other schemas and the objects in those schemas.

2. Open the Indexes node.

   The names of the indexes in the selected schema appear below the Indexes node.

3. Click the name of the index you want to view.

   A tab with the index name appears in the object pane, with the Columns subtab displayed. You can view the index definition on this tab.

See Also:

"About Indexes"

Example: Creating an Index

When you create an index using SQL Developer, you specify one or more table columns to be indexed and the type of index to create.

In this example, you create an index on the `PROD_DESC` column in the `SH.PRODUCTS` table. (The `SH` schema is part of the sample schemas.)

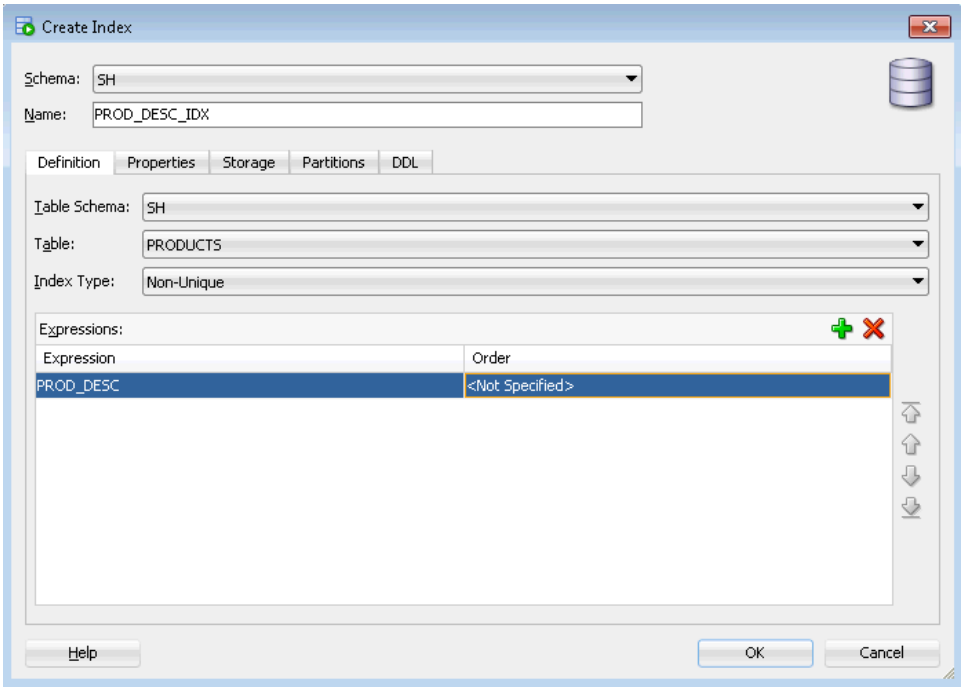
To create a description index on the SH.PRODUCTS table:

1. In SQL Developer, view the tables in the SH schema, by following the instructions in "Viewing Tables".
2. Right-click the PRODUCTS table, and select **Index**, and then **Create Index**.

   The Create Index dialog box appears, with the Definition tab displayed.
3. At the top of the dialog box, confirm that SH appears in the **Schema** field and that PROD_DESC_IDX appears in the **Name** field.
4. On the Definition tab:
   - In the **Table Schema** field, enter SH.
   - In the **Table** field, select PRODUCTS.
   - In the **Index Type** field, select Non-Unique.
   - Click the green check mark item in the Expressions section, and in the row that displays, enter PROD_DESC in the **Expressions** column, and <Not Specified> in the **Order** column.

     If your index were to consist of multiple columns (a concatenated index), then you would add a second column in the Expressions section to add it to the list of columns in the Index list. Then you would use the up arrow and down arrow icons to arrange the index columns in the order you want.

     If the index includes multiple columns, choose either ASC (ascending) or DESC (descending) in the Order field. The value you choose determines the sort order to be used for the index.



5. Click **OK** to create the index.

See Also:

"About Indexes"

Example: Deleting an Index

If you no longer need an index, then you can delete it using SQL Developer.

In this example, you delete the `PROD_DESC_IDX` index that you created previously on the SH.PRODUCTS table in "Example: Creating an Index".

Note:

You cannot delete an index that is currently used to enforce a constraint. You must disable or delete the constraint and then, if the index is not deleted as a result of that action, delete the index.

To delete the description index on the SH.PRODUCTS table:

1. In SQL Developer, view the tables in the `SH` schema, by following the instructions in "Viewing Tables".
2. Right-click the `PRODUCTS` table, and select **Index**, and then **Drop**.

   The Drop dialog box appears, with the Prompts tab displayed.
3. In the **Drop Index** field, select `PROD_DESC_IDX`.
4. Click **Apply**.

   A confirmation message appears.

See Also:

"About Indexes"

Managing Views

The following topics describe how to create and manage views using SQL Developer:

- About Views
- Displaying Views
- Example: Creating a View
- Example: Deleting a View

About Views

Views are customized presentations of data in one or more tables or other views. You can think of them as stored queries. Views do not actually contain data, but instead derive their data from the tables upon which they are based. These tables are referred to as the **base tables** of the view.

Similar to tables, views can be queried, updated, inserted into, and deleted from, with some restrictions. All operations performed on a view actually affect the base tables of the view. Views can provide an additional level of security by restricting access to a predetermined set of rows and columns of a table. They can also hide data complexity and store complex queries.

Many important views are in the SYS schema. There are two types: *static data dictionary views* and *dynamic performance views*. Complete descriptions of the views in the SYS schema are in *Oracle Database Reference*.

Static Data Dictionary Views

The data dictionary views are called **static views** because they change infrequently, only when a change is made to the data dictionary. Examples of data dictionary changes include creating a new table or granting a privilege to a user.

Many data dictionary tables have three corresponding views:

- A DBA_ view displays all relevant information in the entire database. DBA_ views are intended only for administrators.

  An example of a DBA_ view is DBA_TABLESPACES, which contains one row for each tablespace in the database.

- An ALL_ view displays all the information accessible to the current user, including information from the schema of the current user, and information from objects in other schemas, if the current user has access to those objects through privileges or roles.

  An example of an ALL_ view is ALL_TABLES, which contains one row for every table for which the user has object privileges.

- A USER_ view displays all the information from the schema of the current user. No special privileges are required to query these views.

  An example of a USER_ view is USER_TABLES, which contains one row for every table owned by the user.

The columns in the DBA_, ALL_, and USER_ views are usually nearly identical. The USER_ view usually does not have an OWNER column.

Dynamic Performance Views

Dynamic performance views monitor ongoing database activity. They are available only to administrators. The names of dynamic performance views start with the characters V$. For this reason, these views are often referred to as V$ views.

An example of a V$ view is V$SGA, which returns the current sizes of various System Global Area (SGA) memory components.

See Also:

- "Displaying Views"
- "Example: Creating a View"
- "Example: Deleting a View"
- *Oracle Database Concepts*

Displaying Views

You can use SQL Developer to list the views in a specified schema. You can also display the view definitions.

To display views:

1. In the Connections navigator in SQL Developer, navigate to the Views node for the schema that includes the view you want to display.

   If the view is in your own schema, navigate to the Views node in your schema.

   If the view you want to display is in another user's schema, navigate to the Other Users node, expand it, find the name of the schema the view is in, and navigate to the Views node.

   Examples of schema names include `SYS` and `HR`.

   Note:

   You must have the necessary privileges to view other schemas and the objects in those schemas.

2. Open the Views node.

   The list of views in the schema appears.

3. Click the name of the view that you want to display.

   A tab with the view name appears in the object pane, with the Columns subtab displayed. You can view the view definition on this tab.

   See Also:

   "About Views"

Example: Creating a View

In this example, you use SQL Developer to create a view named `king_view`, which uses the `HR.EMPLOYEES` table as its base table. (The `HR` schema is part of the sample schemas.) This view filters the table data so that only employees who report directly to the manager King, whose employee ID is 100, are returned in queries. In an application scenario, this view adds an additional level of security to the `HR.EMPLOYEES` table while providing a suitable presentation of relevant information for manager King.

To create the KING_VIEW view on the HR.EMPLOYEES table:

1. In the Connections navigator in SQL Developer, navigate to the Views node in the HR schema, by following the instructions in "Displaying Views".

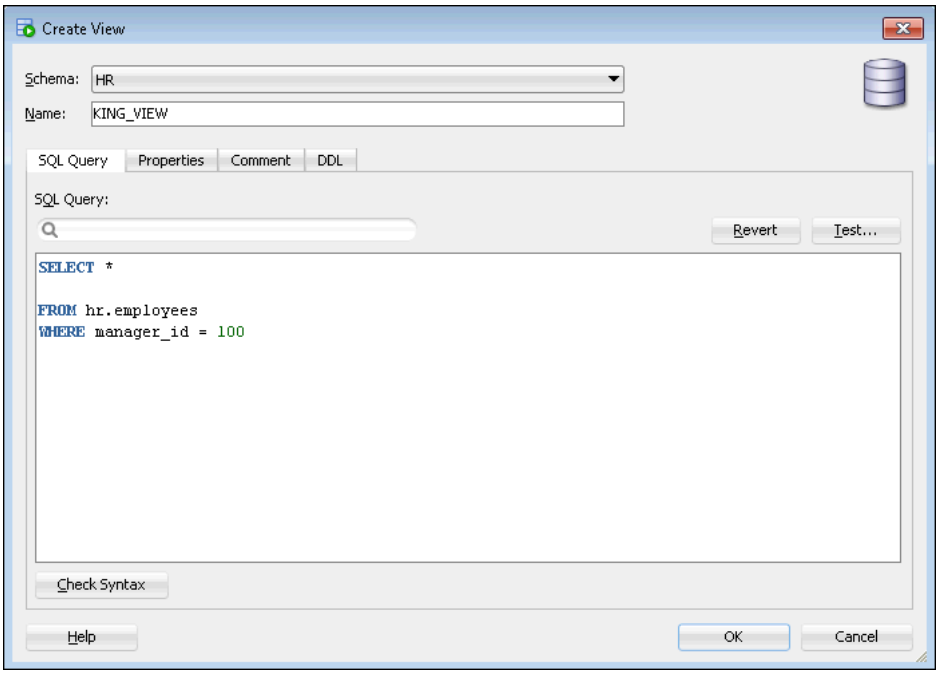
2. Right-click the Views node and select **New View**.

   The Create View dialog box appears, with the SQL Query tab displayed.

3. Enter the following information:

   - In the **Schema** field, select `HR`.
   - In the **Name** field, enter `KING_VIEW`.
   - In the **SQL Query** field, enter the following SQL statement that will be used to create `KING_VIEW`:

     ```
     SELECT * FROM hr.employees
     WHERE manager_id = 100
     ```



4. Click OK.

   The `KING_VIEW` is created and appears in the list of views for the HR schema.

To test the new KING_VIEW view:

1. In the Connections navigator in SQL Developer, navigate to the Views node in the HR schema and find the `KING_VIEW`, by following the instructions in "Displaying Views".

2. Click the `KING_VIEW`.

   A tab with the view name appears in the object pane, with the Columns subtab displayed.

3. Click the Data subtab in the object pane.

   The data selected by the view appears.

4. (Optional) You can also test the view by submitting the following SQL statement in SQL*Plus or SQL Developer:

   ```
   SELECT * FROM hr.king_view
   ```

See Also:

"About Views"

Example: Deleting a View

If you no longer need a view, then you can delete it using SQL Developer.

In this example, you delete the `HR.KING_VIEW` view that you created previously in "Example: Creating a View".

To delete the HR.KING_VIEW view:

1. In the Connections navigator in SQL Developer, navigate to the Views node in the HR schema and find the `KING_VIEW`, by following the instructions in "Displaying Views".
2. Right-click the `KING_VIEW`, and then select **Drop**.

   The Drop dialog box appears.
3. Click **Apply**.

   A confirmation message appears.

See Also:

"About Views"

Managing Program Code Stored in the Database

This section describes your responsibilities as a database administrator (DBA) for program code that is stored in the database. It contains the following topics:

- About Program Code Stored in the Database
- Validating (Compiling) Invalid Schema Objects

About Program Code Stored in the Database

Oracle Database offers the ability to store program code in the database. Developers write program code in PL/SQL or Java, and store the code in schema objects. You, as the DBA, can use SQL Developer to manage program code objects such as:

- PL/SQL packages, procedures, functions, and triggers
- Java source code (Java sources) and compiled Java classes

The actions that you can perform include creating, compiling, creating synonyms for, granting privileges on, and showing dependencies for these code objects. You can also edit and debug PL/SQL code objects using SQL Developer. You access administration pages for these objects by clicking links in the Programs section of the Schema subpage.

Note that creating and managing program code objects is primarily the responsibility of application developers. However, as a DBA you might have to assist in managing

these objects. Your most frequent task for program code objects might be to revalidate (compile) them, because they can become invalidated if the schema objects on which they depend change or are deleted.

Note:

Other types of schema objects besides program code objects can become invalid. For example, if you delete a table, then any views that reference that table become invalid.

See Also:

- *Oracle Database Concepts* for an overview of using PL/SQL and Java for server-side programming
- *Oracle Database 2 Day + Java Developer's Guide* for more information about Java sources and Java classes
- *Oracle Database PL/SQL Language Reference* to learn about PL/SQL code
- *Oracle Database Administrator's Guide* for more information about object invalidation

Validating (Compiling) Invalid Schema Objects

As a database administrator (DBA), you may be asked to revalidate schema objects that have become invalid. Schema objects (such as triggers, procedures, or views) might be invalidated when changes are made to objects on which they depend. For example, if a PL/SQL procedure contains a query on a table and you modify table columns that are referenced in the query, then the PL/SQL procedure becomes invalid. You revalidate schema objects by compiling them.

Note:

It is not always possible to revalidate a schema object that stores program code by compiling it. You may have to take remedial actions first. For example, if a view becomes invalid because a table that it references is deleted, then compiling the view produces an error message that indicates that the table does not exist. You cannot validate the view until you re-create the table.

You can use SQL Developer to run a report that finds invalid schema objects.

To find invalid schema objects:

1. If the Reports navigator does not appear in SQL Developer, choose the **Reports** option from the **View** menu to display the Reports navigator.

   The Reports navigator appears.

2. In the Reports navigator, expand the All Reports node, then expand the Data Dictionary Reports node, then expand the All Objects node, and then click Invalid Objects.

The Select Connection dialog box appears.

3. In the Select Connection dialog box, select the connection to use, or create a new connection.

   If you want the invalid objects report to include information about only the invalid objects in your own schema, use a connection for your own schema.

   If you want the invalid objects report to include information about invalid objects throughout the database, use a connection for a privileged user, such as SYS. In this example, the connection chosen is for the SYS user.

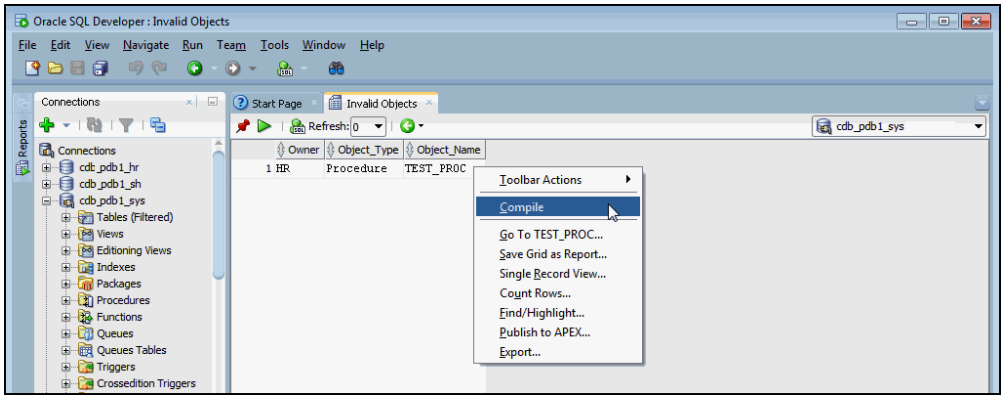
4. Click **OK**.

   The Enter Bind Values dialog box appears.

5. Click **Apply**.

   The Invalid Objects tab appears in the object pane. This tab lists the invalid objects in your schema or in the database (depending on the connection you specified in the Select Connection dialog box).

6. When you right-click the row for a particular invalid object on the Invalid Objects tab, the **Compile** option appears. Select that option to recompile the invalid object.



Remember that it is not always possible to make an object valid by recompiling it. See the Note at the beginning of this section.

See Also:

- *Oracle Database Concepts* for more information about schema object dependencies
- *Oracle Database Administrator's Guide* for information about managing object dependencies

---

Working with Other Schema Objects

In addition to managing tables, indexes, views, and program code with SQL Developer, you can use SQL Developer to manage other schema objects, including the following:

- Sequences

A **sequence** is a database object that generates unique integers. Each time that you query the sequence, it increments its current value by a designated amount and returns the resulting integer. Sequences can be simultaneously queried by multiple users, and each user receives a unique value. For this reason, using a sequence to provide the value for a primary key in a table is an easy way to guarantee that the key value is unique, regardless of the number of users inserting data into the table.

- Synonyms

  A **synonym** is an alias for any schema object, such as a table or view. Synonyms provide an easy way to hide the underlying database structure from an application or a user. Synonyms can be private or public. A public synonym does not have to be qualified with a schema name, whereas a private synonym does, if the user referencing the private synonym is not the synonym owner. For example, consider the following query, issued by a user who has been granted the `SELECT` object privilege on the `HR.EMPLOYEES` table:

  ```
  SELECT   employee_id, salary
  FROM     hr.employees
  ORDER BY salary
  ```

  Now suppose you create a public synonym named `PERSONNEL` as an alias for the `HR.EMPLOYEES` table, and you grant the `SELECT` privilege on the `HR.EMPLOYEES` table to `PUBLIC` (all database users). With the public synonym in place, any user can issue the following simpler query:

  ```
  SELECT   employee_id, salary
  FROM     personnel
  ORDER BY salary
  ```

  The user who created this query did not need to know the name of the schema that contains the personnel data.

  Note:

  If a user owns a table named `personnel`, then that table is used in the query. If no such table exists, then the database resolves the public synonym and uses the `HR.EMPLOYEES` table.

  An additional benefit of synonyms is that you can use the same synonym in a development database as in the production database, even if the schema names are different. This technique enables application code to run unmodified in both environments. For example, the preceding query would run without errors in a development database that had the `EMPLOYEES` table in the `DEV1` schema, if the `PERSONNEL` synonym is defined in the development database to point to the `DEV1` schema.

  Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary. To reference a synonym in a query, you must have privileges on the object to which it points. Synonyms themselves cannot be secured. If you grant object privileges on a synonym to a user, then you are granting privileges on the object to which the synonym points.

- Database links

  A **database link** is a schema object that points to another Oracle database. You use a database link to query or update objects in a remote database. Database links are

used in distributed database environments, which are described in *Oracle Database Administrator's Guide*.

See Also:

- *Oracle Database 2 Day Developer's Guide*
- *Oracle Database SQL Language Reference*

Schemas: Oracle by Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter and includes annotated screenshots.

To view the Managing Schema Objects OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6288,1
```

9

Performing Backup and Recovery

This chapter introduces you to Oracle Database backup and recovery with Oracle Recovery Manager (RMAN). After reading this chapter, you should be familiar with the basic concepts of Oracle Database backup and recovery operations, know how to implement a disk-based backup strategy, and perform simple repairs to database files.

This chapter contains the following sections:

See Also:

- *Oracle Database Backup and Recovery User's Guide* for information about using RMAN to perform advanced backup and recovery operations
- *Oracle Database Backup and Recovery Reference* for the syntax and semantics of RMAN commands

Overview of Database Backup and Recovery

The focus in Oracle Database backup and recovery is on the physical backup of database files, which permits you to reconstruct your database. Oracle Recovery Manager (RMAN), a command-line tool, is the method preferred by Oracle for efficiently backing up and recovering your Oracle database. The files protected by the backup and recovery facilities built into RMAN include data files, control files, server parameter files, and archived redo log files. With these files you can reconstruct your database. RMAN is designed to work intimately with the server, providing block-level corruption detection during backup and restore. RMAN optimizes performance and space consumption during backup with file multiplexing and backup set compression, and integrates with leading tape and storage media products. The backup mechanisms work at the physical level to protect against file damage, such as the accidental

deletion of a data file or the failure of a disk drive. RMAN can also be used to perform point-in-time recovery to recover from logical failures when other techniques such as flashback cannot be used.

Logical backups, such as exporting database objects such as tables or tablespaces, are a useful supplement to physical backups, but cannot protect your whole database. An effective backup strategy must be based on physical backups.

The Oracle Database flashback features provide a range of physical and logical data recovery tools as efficient, easy-to-use alternatives to physical and logical backups. The flashback features enable you to reverse the effects of unwanted database changes without restoring data files from backup.

This section introduces the following flashback features:

- Oracle Flashback Table, which enables you to revert a table to its contents at a time in the recent past
- Oracle Flashback Drop, which enables you to retrieve deleted (dropped) database tables
- Oracle Flashback Database, which enables you to revert the entire database to a past point in time

The first two features operate at the logical level, whereas the last feature operates at the physical level. None of the preceding features requires advance preparation such as creating logical exports to allow for retrieval of your lost data, but Oracle Flashback Database requires the advance preparation of enabling the feature. You can use all of the features while your database is available. *Oracle Database Backup and Recovery User's Guide* discusses the flashback features of Oracle Database at greater length.

Note:

Oracle Flashback Database does not recover missing data files.

Overview of Backing Up and Recovering CDBs and PDBs

When using the multitenant architecture, you can perform backup and recovery operations on a whole multitenant container database (CDB), the root, or one or more pluggable databases (PDBs). The Oracle Recovery Manager (RMAN) commands used to backup and recover CDBs and PDBs are the same as those used for non-CDBs, with minor variations in the syntax. The backup and recovery operations performed on non-CDBs can also be performed on CDBs and PDBs. This includes the following:

- Full backups
- Incremental backups
- Complete recovery
- Point-in-time recovery (PITR)
- Flashback Database (only for CDBs)
- Reporting operations (such as listing backups and cross-checking backups)

Backup and Recovery of CDBs

To perform backup and recovery operations on a whole multitenant container database (CDB), you connect as TARGET to the root. The connection must be established as a user with the SYSDBA or SYSBACKUP privilege.

You can connect to the root in one of the following ways:

- Connect using operating system authentication

  You are connected to the root as the SYS user with the SYSDBA privilege.

- Connect locally as a common user
- Connect as a common user through Oracle Net Services

After you connect to the root, the same commands that are used to perform operations on non-CDBs are used to perform backup and recovery operations on the entire CDB.

See Also:

The following sections in the *Oracle Database Backup and Recovery User's Guide* provide detailed information about backing up and recovering CDBs:

- Connecting as Target to the Root
- Backing Up a Whole CDB
- Performing Complete Recovery of a Whole CDB
- Performing Point-in-Time Recovery of a Whole CDB
- Validating a Whole CDB
- Reporting in CDBs

Backup and Recovery of PDBs

You can perform backup and recovery operations on a single pluggable database (PDB) or on multiple PDBs. Although the Oracle Recovery Manager (RMAN) commands are the same, the syntax used to perform operations on multiple PDBs contains some modifications.

To perform backup and recovery operations on a single PDB, you can connect as TARGET to the PDB or to the root.

- If you connect to the PDB, use the same commands that you would use to backup or recover non-CDBs. For example, to back up a PDB, use the BACKUP DATABASE command.
- If you connect to the root, use the PLUGGABLE DATABASE clause in your RMAN commands.

  The following command backs up the PDB hrpdb when connected to the root:

```
BACKUP PLUGGABLE DATABASE hrpdb;
```

To perform backup and recovery operations on multiple PDBs using a single command, you must connect to the root. Use the PLUGGABLE DATABASE clause followed by the list of PDBs on which you want to perform the operation. The

following example backs up the PDBs `hrpdb`, `salespdb`, and `invpdb` when connected to the root:

```
BACKUP PLUGGABLE DATABASE hrpdb, salespdb, invpdb;
```

Note:

Certain operations are not available when you connect directly to a PDB. See *Oracle Database Backup and Recovery User's Guide* for a list of these operations.

See Also:

The following sections in the *Oracle Database Backup and Recovery User's Guide* provide detailed information about backing up and recovering PDBs:

- Connecting as Target to a PDB
- Connecting as Target to the Root
- Backing Up PDBs with RMAN
- Performing Complete Recovery of PDBs with RMAN
- Performing Point-in-Time Recovery of PDBs
- Validating PDBs
- Reporting in PDBs

Database Backup and Recovery Concepts

To **back up your database** means to make copies of your data files, control file, and archived redo log files (if your database runs in `ARCHIVELOG` mode). **Restoring a database** means copying the physical files that comprise the database from a backup medium, typically disk or tape, to their original or to new locations. **Database recovery** is the process of updating database files restored from a backup with the changes made to the database after the backup by applying incremental backups and redo logs to the restored files.

This section contains the following topics:

- ARCHIVELOG and NOARCHIVELOG Mode
- RMAN Repository
- Image Copies and Backup Sets
- Full Backups and Incremental Backups
- Consistent and Inconsistent Backups
- Media Recovery
- Fast Recovery Area

See Also:

- "Backing Up Your Database"
- *Oracle Database Backup and Recovery User's Guide* for an overview of Oracle Recovery Manager (RMAN) architecture
- *Oracle Database Backup and Recovery User's Guide* for more conceptual details about RMAN backups

ARCHIVELOG and NOARCHIVELOG Mode

One of the important decisions you need to make as a DBA is to determine if the database must be run in `ARCHIVELOG` mode or `NOARCHIVELOG` mode. The mode you choose depends on your availability and reliability requirements. It also impacts the type of backup and recovery operations that you can perform.

In `NOARCHIVELOG` mode, the filled redo log groups that become inactive can be reused. This mode protects the database against instance failure, but not against media failure. In `ARCHIVELOG` mode, filled groups of redo logs are archived. This mode protects the database from both instance and media failure, but may require additional hardware resources.

See Also:

For a detailed discussion about `ARCHIVELOG` and `NOARCHIVELOG` mode, see *Oracle Database Administrator's Guide*

RMAN Repository

Oracle Recovery Manager (RMAN) maintains a record of database files and backups for each database on which it performs operations. This metadata is called the **RMAN repository**. When you request recovery of a database, RMAN uses the repository metadata to choose the most efficient backups needed for this restore and recovery.

The primary location for the RMAN repository for a database is its control file. The importance of this metadata for RMAN is one more reason why protecting your control file is a vital part of your backup strategy. In some installations, a second copy of the RMAN repository is stored in a schema called the **recovery catalog**. The recovery catalog is located in a separate database and can store metadata for multiple target databases.

It is recommended that you use a recovery catalog. Because a recovery catalog stores metadata history for longer than the control file, you can perform a recovery that goes further back in time than the history in the control file. Also, if the target control file and all backups are lost, then the RMAN metadata in the recovery catalog can be used.

See Also:

For more information about creating and managing a recovery catalog, see *Oracle Database Backup and Recovery User's Guide*

Image Copies and Backup Sets

Database backups created by Oracle Recovery Manager (RMAN) are stored as image copies or backup sets.

Image copies are exact byte-for-byte copies of files. You can create an image copy by copying a file at the operating system level. Unlike copying files at the operating system level, however, image copies created through RMAN are recorded in the RMAN repository so that RMAN can use these copies during database restore operations and recovery. RMAN can restore files only if they are recorded in the RMAN repository. RMAN can create image copies only on disk.

Backup sets are logical entities produced by the RMAN BACKUP command. This command can produce one or more backup sets on disk or tape devices. Although image copies cannot use all RMAN features, their advantages are that you can apply incremental backups to them (synthetic full backups) and you can use them directly in place without first copying them, for very fast restores.

Each backup set contains one or more physical files called **backup pieces**. A backup piece stores the backup of one or more database files in a compact RMAN-specific format. One advantage of backup sets is that RMAN uses unused block compression to save space in backing up data files. Only those blocks in the data files that have been used to store data are included in the backup set. Backup sets can also be compressed, encrypted, sent to tape, and use advanced unused-space compression that is not available with datafile copies.

Full Backups and Incremental Backups

A **full backup** of a data file includes all used blocks of the data file. A full backup can be either an image copy or backup set.

An **incremental backup** copies only those blocks in a data file that change between backups. A **level 0 incremental backup**, which copies all blocks in the data file, is used as a starting point for an incremental backup strategy.

A **level 1 incremental backup** copies only images of blocks that have changed since the previous level 0 or level 1 incremental backup. Level 1 backups can be **cumulative**, in which case all blocks changed since the most recent level 0 backup are included, or **differential**, in which case only blocks changed since the most recent level 0 or level 1 incremental backup are included.

Incremental backups at level 0 can be either backup sets or image copies, but incremental backups at level 1 can only be backup sets.

A typical incremental strategy makes level 1 backups at regular intervals such as once each day.

During recovery, Oracle Recovery Manager (RMAN) will automatically apply both incremental backups and redo logs as required, to recover the database to the exact point in time desired.

Consistent and Inconsistent Backups

A backup is either consistent or inconsistent. To make a consistent backup, your database must have been shut down cleanly and remain closed for the duration of the backup. All committed changes are written to the data files during the shut down process, so the data files are in a transaction-consistent state. When you restore your data files from a consistent backup, you can open the database immediately.

If the database is in ARCHIVELOG mode, then you can make inconsistent backups that are recoverable using archived redo log files. Open database backups are inconsistent because the online redo log files contain changes not yet applied to the data files. The online redo log files must be archived and then backed up with the data files to ensure recoverability.

Despite the name, an inconsistent backup is as robust a form of backup as a consistent backup. The advantage of making inconsistent backups is that you can back up your database while the database is open for updates.

Media Recovery

If you restore the archived redo log files and data files, then you must perform media recovery before you can open the database. Any database transactions in the archived redo log files not reflected in the data files are applied to the data files, bringing them to a transaction-consistent state before the database is opened.

Media recovery requires a control file, data files (typically restored from backup), and online and archived redo log files containing changes since the time the data files were backed up. Media recovery is most often used to recover from media failure, such as the loss of a file or disk, or a user error, such as the deletion of the contents of a table.

Media recovery can be a complete recovery or a point-in-time recovery. Complete recovery can apply to individual datafiles, tablespaces, or the entire database. Point-in-time recovery applies to the whole database (and also sometimes to individual tablespaces, with automation help from Oracle Recover Manager (RMAN)).

In a complete recovery, you restore backup data files and apply all changes from the archived and online redo log files to the data files. The database is returned to its state at the time of failure and can be opened with no loss of data.

In a point-in-time recovery, you return a database to its contents at a user-selected time in the past. You restore a backup of data files created before the target time and a complete set of archived redo log files from backup creation through the target time. Recovery applies changes between the backup time and the target time to the data files. All changes after the target time are discarded.

RMAN enables you to perform both a complete and a point-in-time recovery of your database. However, this documentation focuses on complete recovery.

See Also:

- "Performing User-Directed Recovery"
- *Oracle Database Backup and Recovery User's Guide* for more detailed information about point-in-time recovery

Fast Recovery Area

To simplify the management of backup and recovery files, you can create a fast recovery area for your database. The fast recovery area is an Oracle-managed directory, file system, or Oracle Automatic Storage Management disk group that provides a centralized storage location for backup and recovery files. Oracle creates archived logs and flashback logs in the fast recovery area. Oracle Recovery Manager (RMAN) can store its backup sets and image copies in the fast recovery area, and it uses it when restoring files during media recovery. The fast recovery area also acts as a disk cache for tape.

Oracle Database automatically manages this storage, deleting files that are no longer needed. Periodically copying backups to tape frees space in the fast recovery area for other files.

When you issue the RMAN `BACKUP` command without specifying a backup destination, RMAN automatically backs up to the fast recovery area if it is configured.

Oracle recommends that you configure a fast recovery area to simplify backup management. Except as noted, this documentation assumes the use of a recovery area.

See Also:

- "Planning Space Usage and Location for the Fast Recovery Area"
- "Configuring the Fast Recovery Area"

Configuring Your Database for Basic Backup and Recovery

This section explains how to set up your database to take advantage of Oracle suggested backup strategies.

To take maximum advantage of Oracle Database features that automatically manage backup and recovery files and operations, configure your database as follows:

- Use a fast recovery area, which automates storage management for most backup-related files, and specify it as an archived redo log file destination.
- Run the database in `ARCHIVELOG` mode so you can perform online backups and have data recovery options such as complete and point-in-time media recovery.

You must also set several policies governing which files are backed up, what format is used to store backups on disk, and when files become eligible for deletion.

In a multitenant environment, you must connect to the root and configure backup and recovery settings for the whole multitenant container database (CDB). These settings are applicable to the root and to all pluggable databases (PDBs) in the CDB.

This section contains the following topics:

- Planning Space Usage and Location for the Fast Recovery Area
- Configuring Users to Perform Backup and Recovery
- Connecting to the Target Database Using RMAN
- Configuring Recovery Settings
- Configuring Backup Settings

Planning Space Usage and Location for the Fast Recovery Area

Oracle recommends placing the fast recovery area on a separate storage device from the working set of database files. Otherwise, the storage device becomes a single point of failure for your database.

The amount of storage space to allocate for the fast recovery area depends on the size and activity levels of your database and on your recovery objectives. Your objectives dictate what kinds of backups you use, when you make them, and how long to keep them.

This section contains the following topics:

- About the Backup Retention Policy and the Fast Recovery Area
- About the Fast Recovery Area Size

About the Backup Retention Policy and the Fast Recovery Area

Space management in the fast recovery area is governed by a backup retention policy. A retention policy determines when files are obsolete, meaning that they are no longer needed to meet your data recovery objectives.

Retention policies can be based on redundancy of backups or on a recovery window (period of time).

When using a policy based on redundancy, you specify how many full or level 0 backups of each data file and control file that Oracle Recovery Manager (RMAN) keeps. If the number of full or level 0 backups for a specific data file or control file exceeds the redundancy setting, then RMAN considers the extra backups as obsolete.

When using a recovery policy based on a period of time (or window), you specify a time interval in days. Files are obsolete only when they are no longer needed for complete recovery or point-in-time recovery to a system change number (SCN) within the window. Therefore, a recovery retention policy based on a window is recommended.

The default retention policy is a redundancy of 1. Even after files in the fast recovery area are obsolete, they are typically not deleted until space is needed for new files. If space permits, files recently moved to tape remain on disk to avoid restoring them from tape for a recovery. The automatic deletion of obsolete files and files moved to tape from the fast recovery area makes it a convenient archiving destination. Other destinations require manual deletion of logs.

See Also:

- "Configuring Backup Policy Settings"
- *Oracle Database Backup and Recovery User's Guide* for examples of configuring a redundancy-based policy.

About the Fast Recovery Area Size

Oracle Database Backup and Recovery User's Guide explains how to size the fast recovery area. As a general rule, the larger the fast recovery area, the more useful it is. Ideally, the fast recovery area should be large enough for copies of the data files, control files, online redo log files, and archived redo log files needed to recover the database, and also the copies of these backup files that are kept based on the retention policy.

If your backup strategy includes incremental backups, then add enough space to the fast recovery area for these files. If you can move some backups to tape, then you can reduce the size of the fast recovery area. Note that retrieving files from tape causes longer database restore operations and recovery times.

See Also:

Oracle Database Backup and Recovery User's Guide for information about handling a full fast recovery area

Configuring Users to Perform Backup and Recovery

This section contains:

- Credentials Required to Perform Backup and Recovery
- Granting the SYSBACKUP Privilege

Credentials Required to Perform Backup and Recovery

To perform backup and recovery tasks with Oracle Recovery Manager (RMAN), you must connect to the target database as a user with the SYSDBA or SYSBACKUP administrative privilege. The SYSBACKUP privilege encompasses all the privileges required to back up and recover the database. Those privileges are a subset of the privileges included in the SYSDBA administrative privilege.

The following types of users have the SYSBACKUP privilege.

- The SYSBACKUP user.

  When you install the database, the SYSBACKUP user, with the SYSBACKUP privilege, is created automatically.

- Database users to whom you grant the SYSBACKUP privilege.

- Database host users who are members of the OSBACKUPDBA operating system group—for operating system authentication.

  The OSBACKUPDBA group is assigned to a specific operating system group during database installation. For example, on UNIX and Linux systems, the `backupdba` group is typically designated as the OSBACKUPDBA group. Host users in this group can connect to the target database using operating system authentication; they do not need to be defined as a database user.

For the Oracle suggested backup strategy described in this documentation, you use operation system authentication. Consult your operating system documentation for instructions for creating host users and adding them to the OSBACKUPDBA group.

Note:

In previous releases, you needed the SYSDBA administrative privilege to perform backup and recovery tasks. Starting with Oracle Database 12*c*, it is recommended that you use the SYSBACKUP administrative privilege for backup and recovery operations.

Oracle recommends that you do not use the SYSBACKUP user. Instead, create a user and grant the SYSBACKUP privilege to that user.

See Also:

- "Starting SQL*Plus and Connecting to the Database" for more information about operating system authentication
- "Granting the SYSBACKUP Privilege"

Granting the SYSBACKUP Privilege

As a database administrator, you can grant the SYSBACKUP privilege to any database user. When you do so, an entry is made for that user in the password file.

To grant the SYSBACKUP privilege to an existing user:

1. Log into Oracle Enterprise Manager Database Express (EM Express) as user SYS. Be sure to select the `as sysdba` check box on the Login page.

   See "Accessing the Database Home Page."

2. From the Security menu, select **Users**.
3. On the Users page, click the name of the user to whom you want to grant the privilege.
4. On the Account Summary page, in the Privileges tab, click **Edit**.
5. In the Alter Privileges dialog box, scroll to the SYSBACKUP privilege in the left-hand list, select it, and then click the **Right-Arrow** (>) button.

   The SYSBACKUP privilege appears in the right-hand list.

6. Click **OK**.

See Also:

- "Example: Granting Privileges and Roles to a User Account"
- "Password File"

Connecting to the Target Database Using RMAN

To perform backup or recovery operations or to configure backup and recovery settings, you must start the Oracle Recovery Manager (RMAN) client and connect to the target database. A **target database** is the Oracle database that must be backed up or restored using RMAN. Connections to the target database require the SYSDBA or SYSBACKUP administrative privilege.

To connect to the target database:

1. Open a command window.
2. Ensure that the `ORACLE_SID` environment variable is set to the system identifier (SID) of the database.

```
$ ORACLE_SID=prod; export ORACLE_SID
```

3. Do one of the following:

   - To connect as a database user who was granted the SYSBACKUP privilege, enter the following command:

     ```
     rman target '"username as sysbackup"'
     ```

     The single and double quotes are required. Enter the user's password when prompted.

   - To connect with operating system authentication, ensure that you are logged in to the database host as a user who is in the OSBACKUPDBA group (typically the `backupdba` group on UNIX and Linux systems), and enter the following command:

     ```
     rman target /
     ```

     When you do not explicitly specify SYSDBA or SYSBACKUP, you are connected to the target database with the SYSDBA privilege.

Note:

You can start RMAN on one database host and connect to a target database on another host. This is known as connecting to a remote database, and it requires that:

- You supply a net service name in the connect string.
- Your user name be entered in the password file of the remote database to connect as SYSBACKUP.

Connecting to a remote database is outside the scope of this documentation. For more information, see *Oracle Database Backup and Recovery User's Guide*

See Also:

- *Oracle Database Backup and Recovery User's Guide* for information about connecting to multitenant container databases (CDBs) and pluggable databases (PDBs)
- *Oracle Database Backup and Recovery User's Guide* for more information about connecting to the target database
- "Configuring Users to Perform Backup and Recovery"
- "Client Connections" for information about net service names.

Configuring Recovery Settings

This section explains how to configure settings used for instance recovery, media recovery, and fast recovery. It includes the following topics:

- Configuring the Fast Recovery Area
- Enabling Archiving of Redo Log Files
- Enabling Flashback Database

Configuring the Fast Recovery Area

If you did not specify a location for the fast recovery area during installation, the installation process automatically configures a fast recovery area in the Oracle base directory. Oracle recommends, however, that the fast recovery area be located on a separate storage device from the database files.

You can modify the following initialization parameters to relocate the fast recovery area and to adjust its size:

- `DB_RECOVERY_FILE_DEST`

  Specifies the location of the fast recovery area. This can be a file system directory or an Oracle Automatic Storage Management (Oracle ASM) disk group, but not a raw disk.

- `DB_RECOVERY_FILE_DEST_SIZE`

  Specifies the size of the fast recovery area, in bytes.

The `DB_RECOVERY_FILE_DEST_SIZE` parameter must be set before the `DB_RECOVERY_FILE_DEST` parameter.

You can set these parameters without having to shut down and restart the database. In an Oracle Real Application Clusters (Oracle RAC) database, all instances must have the same values for these initialization parameters. The location must be on a cluster file system, Oracle ASM, or a shared directory.

To configure the fast recovery area:

Assume that you want to want to place the fast recovery area in the directory `/u02/oracle/fra` and you want its size to have an upper limit of 10 GB.

1. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."
2. Specify the size of the fast recovery area using the following command:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE = 10G;
   ```

3. Specify the location of the fast recovery area using the following command:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST = '/u02/oracle/fra';
   ```

See Also:

- "Fast Recovery Area"
- "About the Fast Recovery Area Size"

Enabling Archiving of Redo Log Files

To back up the database while it is open, or to be able to perform complete or point-in-time media recovery, you must enable the archiving of redo log files. To do so, you place the database in `ARCHIVELOG` mode. You can determine if archiving of redo logs is enabled for the target database using the following query:

```
SELECT LOG_MODE FROM V$DATABASE;
```

If you do not specify a destination to which the database should write archived log files, the database writes them to the fast recovery area. You can specify a different destination, or you can specify that multiple copies of each archived log file be written, each to a different destination. Redundant copies help ensure that archived log files are always available in the event of a failure at one of the destinations.

The following procedure assumes that you want to place archived log files in the directory `/u02/oracle/logfiles`, and redundant copies of archived log file in the directory `/u03/oracle/logfiles`. The redundant copies are optional.

Warning:

You must ensure that there is sufficient disk space at all times for archived log file destinations. If the database encounters a disk full error as it attempts to archive a log file, a fatal error occurs and the database stops responding. You can check the alert log for a disk full message.

To enable archiving of redo log files:

1. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."
2. Shut down the database.

   ```
   SHUTDOWN IMMEDIATE;
   ```

3. Back up the database.

   It is recommended that you always back up a database before making any major change to the database.

   See "Performing a Whole Database Backup."

4. Start the instance and mount the database (do not open the database). To enable archiving, the database must be mounted but not open.

   ```
   STARTUP MOUNT;
   ```

5. Enter the following command to set the first archived log file destination:

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = 'LOCATION=/u02/oracle/logfiles';
   ```

   Note:

   The directory must exist.

   ---

6. (Optional) Enter the following command to set the second archived log file destination:

   ```
   ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'LOCATION=/u03/oracle/logfiles';
   ```

7. Change the database archiving mode and then open the database for normal operations.

   ```
   ALTER DATABASE ARCHIVELOG;
   ALTER DATABASE OPEN;
   ```

8. Shutdown the database.

```
SHUTDOWN IMMEDIATE;
```

9. Back up the database.

   Because changing the archiving mode updates the control file, it is recommended that you create a new backup.

   See "Performing a Whole Database Backup."

10. Start up the database normally.

```
STARTUP;
```

Note:

You cannot use backups from *before* the switch to `ARCHIVELOG` mode to restore and recover the database to a point in time *after* the switch. Thus, if you do not *immediately* make a backup after switching, then you are running your database without a valid backup. See "Performing and Scheduling Backups Using RMAN" to learn how to make database backups.

See Also:

- *Oracle Database Administrator's Guide* for more information about setting initialization parameters for archived log file destinations
- "Media Recovery"
- "Monitoring General Database State and Workload" for information about the alert log

---

Enabling Flashback Database

To revert the entire database to a prior point in time, you can either revert the entire database to a prior point in time by restoring a backup and doing point-in-time recovery, or you can enable Flashback Database. When you enable Flashback Database, the database generates flashback logs in the fast recovery area. These logs are used to flash back the database to a specified time. During usual operation, the database occasionally logs images of data blocks to the flashback logs. The database automatically creates, deletes, and resizes flashback logs.

Use the following command to check if Flashback Database is enabled for your target database:

```
SELECT FLASHBACK_ON FROM V$DATABASE;
```

To enable Flashback Database:

1. Ensure that you configure a fast recovery area and that the database is running in `ARCHIVELOG` mode.

   See "Configuring the Fast Recovery Area" and "Enabling Archiving of Redo Log Files."

2. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."

3. Optionally, specify the length of the desired flashback window (in minutes) by setting the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter.

   The default value for this parameter is 1440 minutes, which is one day. The following command specifies that the flashback window must be 3 days.

   ```
   ALTER SYSTEM SET DB_FLASHBACK_RETENTION_TARGET=4320;
   ```

4. Enable the Flashback Database feature for the whole database using the following command:

   ```
   ALTER DATABASE FLASHBACK ON;
   ```

You can also execute the commands in this section by connecting to the target database using SQL*Plus instead of RMAN.

See Also:

- "Rewinding a Database Using Oracle Flashback Database"
- *Oracle Database Backup and Recovery User's Guide* for more details about configuring a fast recovery area

Configuring Backup Settings

You can configure several backup-related settings and policies. For example, you can determine how backups are stored, which data is backed up, and how long backups are retained. You can also configure settings to optimize backup performance.

This section contains the following topics:

- Configuring Backup Device Settings
- Configuring Backup Policy Settings
- Configuring Automatic Backups for the Control File and Server Parameter File
- Enabling Block Change Tracking

Configuring Backup Device Settings

For disk-based backups, you can configure the default format for backups, the location on disk where backups are stored, whether backup tasks run in parallel, and whether backups are compressed.

For tape backups, you can configure settings such as the number of tape drives and whether backups are compressed. On most platforms, you must integrate a media manager with the Oracle database to use sequential media for storage.

You can use Oracle Secure Backup, which supports both database and file system backups to tape, as your media manager. Oracle Secure Backup provides the same services for Oracle Recovery Manager (RMAN) as other third-party SBT interfaces. This section assumes that you make only disk backups.

Use the RMAN `CONFIGURE` command to configure backup device settings such as the default backup type, disk location to which database files are backed up, and parallelism. Use the RMAN `SHOW ALL` command to view the currently configured settings.

To configure backup device settings:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Specify that the default device used to store backups is disk. The following command directs RMAN to store backups to disk.

   ```
   CONFIGURE DEFAULT DEVICE TYPE TO DISK;
   ```

3. Specify that the backups must be stored on disk in the form of backup sets. Also set the parallelism to 1.

   ```
   CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO BACKUPSET PARALLELISM 1;
   ```

   Backups on disk can be stored in the form of backup sets or image copies. Image copies are exact copies of database files. They are not stored in an RMAN-specific format and can be used as-is to perform recovery. Backup sets use an RMAN-specific format. With backup sets, RMAN uses unused block compression to save space by backing up only the blocks that contain data. RMAN can also encrypt backups and create incremental backups.

Note:

For more information about image copies and backup sets, see *Oracle Database Backup and Recovery User's Guide*

See Also:

Oracle Database Backup and Recovery User's Guide for information about setting tape as the default device

---

Configuring Backup Policy Settings

You can set the backup policies that govern control file and server parameter file backups, tablespaces to exclude from an entire database backup, and the backup retention policy.

To configure the backup policy settings:

1. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."
2. Configure RMAN to automatically backup the control files and the server parameter as described in "Configuring Automatic Backups for the Control File and Server Parameter File".
3. Configure backup optimization to save space in the fast recovery area. Optimization excludes unchanged files, such as read-only files and offline data files, that were previously backed up.

   ```
   CONFIGURE BACKUP OPTIMIZATION ON;
   ```

4. Configure the retention policy to specify how long the backups and archived redo logs must be retained for media recovery.

The following command specifies that the backups and archived logs must be retained for 31 days.

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 31 DAYS;
```

5. Specify that archived logs can be automatically deleted only when they have been backed up to tape or are obsolete based on the retention policy by using the following command.

   ```
   CONFIGURE ARCHIVELOG DELETION POLICY TO NONE;
   ```

6. Enable block change tracking as described in "Enabling Block Change Tracking".

7. Optionally, specify the tablespaces that must be excluded from backups.

   Excluding certain tablespaces from backups, such as read-only tablespaces or tablespaces that contain temporary or test data, enables you to save space.

   The following command excludes the tablespace `example` from backups.

   ```
   CONFIGURE EXCLUDE FOR TABLESPACE example;
   ```

See Also:

Oracle Database Backup and Recovery User's Guide for information about other methods of configuring the retention policy

Configuring Automatic Backups for the Control File and Server Parameter File

You can configure Oracle Recovery Manager (RMAN) to automatically backup the control file and server parameter file with every backup. This is referred to as an **autobackup**. The server parameter file and control file are critical to the database and RMAN. Creating automatic backups of the control file enables RMAN to recover the database even if the current control file and server parameter file are lost. The control file and server parameter file are relatively small compared to typical data files and, therefore, backing them up frequently results in relatively little storage overhead.

If the database runs in `ARCHIVELOG` mode, then an autobackup is also taken whenever the database structure metadata in the control file changes.

Note:

For multitenant container databases (CDBs), the control file and server parameter file autobackups are performed by default.

To configure automatic backups for the control file and server parameter file:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."

2. Enter the following command:

   ```
   CONFIGURE CONTROLFILE AUTOBACKUP ON;
   ```

RMAN uses a default format to assign names for these backups.

See Also:

Oracle Database Backup and Recovery User's Guide for more information about configuring the format for backups

Enabling Block Change Tracking

Block changing tracking improves the performance of incremental backups by recording changed blocks in the block change tracking file. During an incremental backup, instead of scanning all data blocks to identify which blocks have changed, RMAN uses this file to identify the changed blocks that need to be backed up.

You can enable block change tracking when the database is either open or mounted. This section assumes that you intend to create the block change tracking file as an Oracle Managed File in the database area, which is where the database maintains active database files such as data files, control files, and online redo log files.

To determine if block change tracking is enabled, check the `STATUS` and `FILENAME` columns in the `V$BLOCK_CHANGE_TRACKING` view, using the following statement from the SQL or RMAN prompt:

```
SELECT status, filename FROM V$BLOCK_CHANGE_TRACKING;
```

To enable block change tracking:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Determine the current location of the database data files by submitting the following query:

   ```
   RMAN> SELECT NAME FROM V$DATAFILE;

   NAME
   ------------------------------------------------
   /u01/app/oracle/oradata/orcl/system01.dbf
   /u01/app/oracle/oradata/orcl/example01.dbf
   /u01/app/oracle/oradata/orcl/sysaux01.dbf
   /u01/app/oracle/oradata/orcl/undotbs01.dbf
   /u01/app/oracle/oradata/orcl/users01.dbf
   ```

   In this example, the query results show that data files are stored in the file system in the directory /u01/app/oracle/oradata/orcl. Data files might also be stored in an Oracle Automatic Storage Management disk group.

3. Set the `DB_CREATE_FILE_DEST` initialization parameter to specify the location where new database files, including the block change tracking file, must be stored. You can specify the same directory shown in query results from the previous step, with the final portion of the path—the database SID—stripped, as shown in the following example, or designate a new directory. Any directory that you specify must have the write permission for the Oracle software owner.

   The following command specifies that new database files must be stored in the directory /u01/app/oracle/oradata/:

   ```
   ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/app/oracle/oradata';
   ```

4. Enable block change tracking for the database using the following command:

   ```
   ALTER DATABASE ENABLE BLOCK CHANGE TRACKING;
   ```

See Also:

- *Oracle Database Reference* for more information about the `V$BLOCK_CHANGE_TRACKING` view.
- *Oracle Database Reference* for more information about the `DB_CREATE_FILE_DEST` initialization parameter

Backing Up Your Database

This section describes how to back up your database with Oracle Recovery Manager (RMAN). The Oracle suggested strategy for disk-only backups provides efficient daily backup of the database. This strategy enables you to quickly return your database to its state at any point during the previous 24 hours. For more flexible backup options, see *Oracle Database Backup and Recovery User's Guide*.

This section contains the following topics:

- Additional Backup Concepts
- Performing and Scheduling Backups Using RMAN
- Displaying Backups Stored in the RMAN Repository
- Validating Backups and Testing Your Backup Strategy

See Also:

"Database Backup and Recovery Concepts"

Additional Backup Concepts

This section contains the following topics:

- Incrementally Updated Backups: Rolling Forward Image Copies of Data Files
- Backup Tags

Incrementally Updated Backups: Rolling Forward Image Copies of Data Files

Oracle Recovery Manager (RMAN) enables you to apply level 1 incremental backups to an older image copy of your data files. You can roll forward the copy to the point in time of the most recent level 1 incremental backup. All blocks changed since the image copy was created are overwritten with their new contents as of the time of the last level 1 incremental backup. The effect is to roll forward the file in time, so that its contents are equivalent, for the purposes of database recovery, to an image copy of the data file made at the time of the last incremental level 1 backup.

Incorporating incrementally updated backups into your backup strategy shortens expected recovery times. Media recovery to the present time or to a point in time in the recent past can begin at the time of the last level 1 backup applied, rather than at the time of the last full database backup.

See Also:

"Using the Oracle Suggested Backup Strategy"

Backup Tags

A **tag** is a text string that identifies a backup, either uniquely or as part of a group of backups. All Oracle Recovery Manager (RMAN) backups, including incremental backups, are labeled with a tag. For example, if you performed a full database backup every Saturday, then you could use the tag `FULL_SAT` to identify this backup.

You can use tags to refer to specific backups in RMAN commands. For example, you could issue a command to move the latest `FULL_SAT` backup to tape. If you do not specify a tag, then RMAN creates a unique tag automatically.

Because you can use tags to refer to different groups of backups, you can create different routines in your backup strategy that do not interfere with each other. When you schedule a backup job and give the job a name, the job name is the tag.

Performing and Scheduling Backups Using RMAN

Oracle Recovery Manager (RMAN) enables you to perform the different types of backups that are required by your backup strategy. This section discusses creating a whole database backup.

You can also individually back up data files, control files, and archived redo log files. You can use some advanced RMAN features such as encrypting backups. For more information about these topics, see *Oracle Database Backup and Recovery User's Guide*.

See Also:

Oracle Database Backup and Recovery User's Guide for information about backing up multitenant conainer databases (CDBs) and pluggable databases (PDBs)

This section contains the following topics:

- Performing a Whole Database Backup
- Using the Oracle Suggested Backup Strategy
- About the Oracle Suggested Backup Strategy and Retention
- Scheduling Miscellaneous Backup Tasks

Performing a Whole Database Backup

Whole backups of a database include the complete contents of all data files of the database, plus the control file, archived redo log files, and server parameter file. With these files, you can perform a complete recovery.

While whole database backups can be an important element in your overall backup strategy, they are also a required step in some situations, such as when you enable or disable `ARCHIVELOG` mode (see "Configuring Recovery Settings"). This section explains how to make whole database backups, both offline and online, to disk. Typically, you want to perform online backups to maximize database availability.

To perform a whole database backup when the database is open:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Ensure that your database is in `ARCHIVELOG` mode as described in "Enabling Archiving of Redo Log Files."

   You can make online backups only if your database is running in `ARCHIVELOG` mode.
3. Back up the database, along with archived redo logs by using the following command:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG;
   ```

This backup is created on the default device that you configured for storing backups. If you did not configure a default device, then the backup is created in the fast recovery area. RMAN uses a default format while naming the backup sets that comprise the backup.

To perform a whole database backup when the database is closed:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Shut down the database and then mount the database using the following commands.

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ```
3. Back up the database using the following command:

   ```
   BACKUP DATABASE;
   ```

   While backing up a database that is closed, you need not back up the archived log files because the database is in a consistent state at the time of backup. Therefore, no media recovery is required if you restore the database from this backup.
4. Open the database after the backup is complete.

   ```
   ALTER DATABASE OPEN;
   ```

See Also:

- *Oracle Database Backup and Recovery User's Guide* for information about performing backups of multitenant container databases (CDBs) and pluggable databases (PDBs)
- "Displaying Backup Reports"
- "Fast Recovery Area"

Using the Oracle Suggested Backup Strategy

The Oracle suggested backup strategy is a scheduled disk backup strategy that protects your data and provides efficient recoverability to any point in the user-specified recovery window (time period). It leverages the incrementally updated

backup features to provide faster backups than whole database backups, and faster recoverability than is possible by applying multiple incremental backups to the last full backup.

Complete the following tasks, detailed in this section, to schedule a daily backup that implements the Oracle suggested backup strategy:

- Task 1 - Preparing to Use the Oracle Suggested Backup Strategy
- Task 2 - Creating the Backup Script—UNIX and Linux
- Task 3 - Testing the Backup Script
- Task 4 - Scheduling the Daily Backup—UNIX and Linux

This section also contains the following topic:

- About the Oracle Suggested Backup Strategy

See Also:

"Incrementally Updated Backups: Rolling Forward Image Copies of Data Files"

About the Oracle Suggested Backup Strategy

The Oracle suggested backup strategy is based on incrementally updated backups. This strategy starts with an image copy of each data file and then rolls forward the image copies each day by applying an incremental level 1 backup.

For each data file, the strategy calls for backups as follows:

- At the beginning of day 1 of the strategy (the time the first scheduled job actually runs), Oracle Recovery Manager (RMAN) creates an incremental level 0 image copy. It contains the data file contents at the beginning of day 1.

  If a recovery is required, then the archived redo log files from day 1 can be used to recover to any point during day 1.

- At the beginning of day 2, RMAN creates a differential incremental level 1 backup that contains the blocks changed during day 1.

  If a recovery is required, then RMAN can apply this incremental level 1 to roll forward the level 0 backup to the beginning of day 2. RMAN can use archived redo log files to recover to any point during day 2.

- At the beginning of each day *n* for day 3 and onward, RMAN applies the level 1 backup from the beginning of day *n*-1 to the level 0 backup. This action brings the data file copy to its state at the beginning of day *n*-1. Then, RMAN creates a new level 1 backup that contains the blocks changed during day *n*-1.

  If a recovery is required, then RMAN can apply this incremental level 1 backup to the data file rolled forward on day *n*-1 to the beginning of day *n*. RMAN can use archived redo log files to recover the database to any point during day *n*.

In this Oracle suggested backup strategy, the data file image copies and the level 1 incremental backups share the same tag. You can safely implement other backup strategies without interfering with the Oracle suggested backup strategy.

Oracle suggested backup strategies also use tape backups in addition to disk backups, but these are beyond the scope of this section.

Task 1 - Preparing to Use the Oracle Suggested Backup Strategy

To use the Oracle suggested backup strategy, ensure that:

- The database is in `ARCHIVELOG` mode.
- The fast recovery area size is configured, or a default device for storing backups is configured.
- You have added a database host user to the OSBACKUPDBA operating system group, for operating system authentication.

See Also:

"Configuring Your Database for Basic Backup and Recovery"

Task 2 - Creating the Backup Script—UNIX and Linux

This backup script implements the Oracle suggested backup strategy, enabling quick recovery to any time in the preceding 24 hours. This script can be used to back up a non-CDB or a whole multitenant container database (CDB).

To create the backup script for UNIX and Linux:

- Start a text editor and create and save a file with the following contents. Save the file in a directory that is accessible to the Oracle Database software and on which the Oracle software owner has the read permission.

Note:

In the following script, substitute the correct values for your installation for the `ORACLE_HOME` and `ORACLE_SID` environment variables.

```
#!/bin/sh
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1
export ORACLE_SID=orcl
PATH=$ORACLE_HOME/bin:$PATH
rman <<EOF
connect target /
RUN {
 ALLOCATE CHANNEL disk_iub DEVICE TYPE DISK;
 RECOVER COPY OF DATABASE WITH TAG daily_iub;
 BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG daily_iub DATABASE;
}
exit
EOF
```

Task 3 - Testing the Backup Script

It is recommend that you run the script manually, to check for errors, before scheduling it. Your manual run of the script will start day one of the strategy, creating an incremental level 0 image copy of all datafiles.

To test the backup script:

1. Log in to the database host as a user who is a member of the OSBACKUPDBA operating system group (typically the `backupdba` group).

2. In a command window, enter the following command:

   ```
   full-script-path
   ```

   where *full-script-path* is the full path and file name of the script you created in Task 2.

   For example, if your script is in the file /u01/app/oracle/rman/daily_backup.sh, then enter this command:

   ```
   /u01/app/oracle/rman/daily_backup.sh
   ```

The script starts Oracle Recovery Manager (RMAN), which starts the backup. The output from RMAN includes warning messages similar to the following:

```
...
no copy of datafile 1 found to recover
no copy of datafile 2 found to recover
...
no parent backup or copy of datafile 1 found
no parent backup or copy of datafile 2 found
...
```

These messages are normal for the first run of the script.

Note:

For the second run of the script, the output includes only these warning messages:

```
no copy of datafile 1 found to recover
no copy of datafile 2 found to recover
...
```

Again, these messages are normal. For the third and subsequent script runs, no further warning messages are output.

Task 4 - Scheduling the Daily Backup—UNIX and Linux

The following procedure uses the `cron` utility to schedule daily database backups at 2:00 a.m.

To schedule the Oracle-suggested disk backup strategy:

1. Ensure that you are logged in to the database host as a user who is a member of the OSBACKUPDBA operating system group (typically the `backupdba` group).

   The `cron` job will run as this host user.

2. Start a text editor, and create and save a file with the following contents into your home directory. Name the file .crontab. (Note the period at the front of the file name.)

   ```
   MAILTO=first.last@example.com
   # MI HH DD MM DAY CMD
     00  2  *  *  *  full-script-path
   ```

   where *full-script-path* is the full path and file name of the script you created in Task 2.

   For example, if the script is in the file /u01/app/oracle/rman/daily_backup.sh, then the .crontab file must contain:

```
MAILTO=first.last@example.com
# MI HH DD MM DAY CMD
  00  2  *  *  *  /u01/app/oracle/rman/daily_backup.sh
```

Note:

Supply the desired e-mail address in the `MAILTO` line. This line is optional. Content written to stdout by the `cron` job is e-mailed to this address at the completion of the job.

3. In a command window, change directory to your home directory and enter the following command:

```
crontab .crontab
```

This creates a crontab file for this user from the contents of .crontab.

Caution:

The existing crontab file for this user is overwritten. If you want to preserve the contents of this file and add this new job, use this command, which enables you to edit the existing file:

```
crontab -e
```

4. (Optional) Check the contents of the crontab file for this user with the following command:

```
crontab -l

MAILTO=first.last@example.com
# MI HH DD MM DAY CMD
  00  2  *  *  *  /u01/app/oracle/rman/daily_backup.sh
```

See Also:

Your operating system documentation for a description of the crontab command and crontab files

About the Oracle Suggested Backup Strategy and Retention

When using the Oracle suggested backup strategy, the retention is dictated by the recovery and not by the configured retention. In order to get retention beyond 24 hours, you must change the RECOVER statement to something like:

```
RECOVER COPY OF DATABASE WITH TAG 'ORA_OEM_LEVEL_0' UNTIL TIME "SYSDATE-4";
```

The configured retention is not honored for either the retention or obsolete settings. So when using the Oracle suggested backup strategy, Oracle recommends that the default setting remains unchanged to avoid confusion:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
```

Scheduling Miscellaneous Backup Tasks

In addition to implementing the Oracle suggested backup strategy described in "Using the Oracle Suggested Backup Strategy," you can use customized backup strategies that

back up certain parts of your database. Customized strategies include backing up selected tablespaces, datafiles, and archived redo logs. Create a script that contains the commands required to implement your customized backup task and then schedule this backup task using the `cron` utility.

See Also:

Oracle Database Backup and Recovery User's Guide for information about performing customized backups

Displaying Backups Stored in the RMAN Repository

Use the `LIST` command to view information about backups stored in the Oracle Recovery Manager (RMAN) repository. The information includes backups of data files, individual tablespaces, archived redo log files, and control files. You can also use this command to display information about expired and obsolete backups.

See Also:

Oracle Database Backup and Recovery User's Guide for more information about the `LIST` command

The syntax used to display backups of multitenant container databases (CDBs) and pluggable databases (PDBs), which has minor variations from that used for non-CDBs, is described in *Oracle Database Backup and Recovery User's Guide*.

To display all backups:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Use the `LIST` command to display a summary of all the backups, both backup sets and image copies.

```
LIST BACKUP SUMMARY;

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ---------- ---
12      B  F  A DISK        28-MAR-12       1       1       NO
TAG20120328T051810
13      B  F  A DISK        28-MAR-12       1       1       NO
TAG20120328T051811
14      B  F  A DISK        28-MAR-12       1       1       NO
TAG20120328T051921
15      B  F  A DISK        28-MAR-12       1       1       NO
TAG20120328T051936
16      B  F  A DISK        28-MAR-12       1       1       NO
TAG20120328T052241
```

To display selected backups:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."

2. Use the `LIST BACKUP` or `LIST COPY` command to display the specified backups, both backup sets and image copies.

   - For example, to list the backups of a particular datafile:

     ```
     LIST BACKUP OF DATAFILE 3;
     LIST COPY OF DATAFILE '/orcl/oradata/trgt/system01.dbf';
     ```

   - To display backups sorted by the type of database file:

     ```
     LIST BACKUP BY FILE;

     List of Datafile Backups
     ========================
     File Key     TY LV S Ckp SCN    Ckp Time  #Pieces #Copies Compressed Tag
     ---- ------- -  -- - ---------- --------- ------- ------- ---------- ---
     1    14      B  F  A 723546     28-MAR-12 1       1       NO
     TAG20120328T051921
     2    14      B  F  A 723546     28-MAR-12 1       1       NO
     TAG20120328T051921
     3    14      B  F  A 723546     28-MAR-12 1       1       NO
     TAG20120328T051921
     4    14      B  F  A 723546     28-MAR-12 1       1       NO
     TAG20120328T051921
     5    14      B  F  A 723546     28-MAR-12 1       1       NO
     TAG20120328T051921

     List of Control File Backups
     ============================
     CF Ckp SCN Ckp Time  BS Key  S #Pieces #Copies Compressed Tag
     ---------- --------- ------- - ------- ------- ---------- ---
     723835     28-MAR-12 16      A 1       1       NO
     TAG20120328T052241
     723557     28-MAR-12 15      A 1       1       NO
     TAG20120328T051936
     723490     28-MAR-12 13      A 1       1       NO
     TAG20120328T051811
     ```

Validating Backups and Testing Your Backup Strategy

As part of your backup strategy, you should periodically check whether your backups are intact and can be used to meet your recoverability objectives. You can validate your backups in the following ways:

- Select specific backup sets or image copies in Oracle Recovery Manager (RMAN) and validate them. This technique indicates if a backup exists and can be restored. For this form of validation, use the steps described in "Validating Selected Backups."
- Specify database files and let RMAN select backups to use when restoring those files, as it would for an actual restore operation. This technique ensures that your available backups are sufficient to restore the database. For this form of validation, use the steps described in "Validating Backups for Restore Operations."

Note:

Validating backups stored on tape can be time-consuming because the entire backup is read from tape.

You can perform both forms of validation using RMAN. You should incorporate both forms of validation into your backup strategy to ensure that your recoverability goals are met by your available backups.

See Also:

Oracle Database Backup and Recovery User's Guide for information about validating backups of multitenant container databases (CDBs) and pluggable databases (PDBs)

Validating Selected Backups

Validating specific backups checks whether these backups exist and can be restored. It does not test whether the set of available backups meet your recoverability goals. For example, image copies of data files for several tablespaces from your database may exist, each of which can be validated. If there are some tablespaces for which no valid backups exist, however, then you cannot restore and recover the database.

To validate selected backups:

1. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."
2. Validate the required backup.

   The following `VALIDATE` command validates the datafile `users_02.dbf`.

```
VALIDATE DATAFILE '/ora112/oradata/users_02.dbf';

Starting validate at 27-MAR-12
using channel ORA_DISK_1
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
input datafile file number=00020
name=/ora112/oradata/users_02.dbf
channel ORA_DISK_1: validation complete, elapsed time: 00:00:01

List of Datafiles
=================
File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
20   OK     0              248          256             618976

File Name: /ora112/oradata/users_02.dbf
Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              8

Finished validate at 27-MAR-12
```

When you suspect that one or more backup pieces in a backup set are missing or have been damaged, use the `VALIDATE BACKUPSET` command to validate the backup set.

See Also:

Oracle Database Backup and Recovery User's Guide for more information about validating backups

Validating Backups for Restore Operations

You can test whether a sufficient set of backups exists that can be used to restore the specified database files. After you specify which tablespaces to restore and, possibly, a

time as of which to restore them, Oracle Recovery Manager (RMAN) selects a set of backups that contain the needed data. RMAN reads the selected backups in their entirety to confirm that they are not corrupt, but does not produce output files.

Validating the restoration of files tests whether the file can be restored given the available backups, but it does not test whether all backups of the specified object are valid.

To verify whether specified database files can be restored:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Run the `RESTORE ... VALIDATE` command to determine if the required database files can be restored.

   - To determine if the whole database can be restored:

     ```
     RESTORE VALIDATE DATABASE;

     Starting restore at 29-MAR-12
     using target database control file instead of recovery catalog
     allocated channel: ORA_DISK_1
     channel ORA_DISK_1: SID=129 device type=DISK
     channel ORA_DISK_1: scanning datafile copy /ade/b/191802369/oracle/work/
     orcva/RDBMS/datafile/o1_mf_tbs_3_7q60nj4y_.dbf
     channel ORA_DISK_1: starting validation of datafile backup set
     channel ORA_DISK_1: reading from backup piece /ade/b/191802369/oracle/
     work/orcva/RDBMS/backupset/2012_03_28/
     o1_mf_nnndf_TAG20120328T051921_7q60g9oc_.bkp
     channel ORA_DISK_1: piece handle=/ade/b/191802369/oracle/work/orcva/RDBMS/
     backupset/2012_03_28/o1_mf_nnndf_TAG20120328T051921_7q60g9oc_.bkp
     tag=TAG20120328T051921
     channel ORA_DISK_1: restored backup piece 1
     channel ORA_DISK_1: validation complete, elapsed time: 00:00:04
     Finished restore at 29-MAR-12
     ```

   - To determine if a specified tablespace can be restored:

     ```
     RESTORE TABLESPACE example VALIDATE;

     Starting restore at 29-MAR-12
     using channel ORA_DISK_1
     channel ORA_DISK_1: starting validation of datafile backup set
     channel ORA_DISK_1: reading from backup piece /ade/b/191802369/oracle/
     work/orcva/RDBMS/backupset/2012_03_28/
     o1_mf_nnndf_TAG20120328T051921_7q60g9oc_.bkp
     channel ORA_DISK_1: piece handle=/ade/b/191802369/oracle/work/orcva/RDBMS/
     backupset/2012_03_28/o1_mf_nnndf_TAG20120328T051921_7q60g9oc_.bkp
     tag=TAG20120328T051921
     channel ORA_DISK_1: restored backup piece 1
     channel ORA_DISK_1: validation complete, elapsed time: 00:00:01
     Finished restore at 29-MAR-12
     ```

   - To determine if a datafile can be restored to a specified SCN:

     ```
     RESTORE DATAFILE 1 VALIDATE UNTIL SCN 23456;

     Starting restore at 29-MAR-12
     using channel ORA_DISK_1
     channel ORA_DISK_1: starting validation of datafile backup set
     channel ORA_DISK_1: reading from backup piece /ade/b/191802369/oracle/
     work/orcva/RDBMS/backupset/2012_03_28/
     o1_mf_nnndf_TAG20120330T044454_7qc75qyd_.bkp
     channel ORA_DISK_1: piece handle=/ade/b/191802369/oracle/work/orcva/RDBMS/
     backupset/2012_03_28/o1_mf_nnndf_TAG20120330T044454_7qc75qyd_.bkp
     tag=TAG20120330T044454
     ```

```
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: validation complete, elapsed time: 00:00:03
Finished restore at 29-MAR-12
```

See Also:

- *Oracle Database Concepts* for datafile recovery concepts
- *Oracle Database Backup and Recovery User's Guide* to learn how to use the `RESTORE ... VALIDATE` command

Displaying Backup Reports

Backup reports contain summary and detailed information about past backup jobs run by Oracle Recovery Manager (RMAN). The view `V$RMAN_BACKUP_JOB_DETAILS` contains information about backup jobs run by RMAN. This view contains information such as the time taken for the backup, when a job started and finished, and what type of backup was performed, and the status of the backup job.

To display backup reports:

Use the following query to display the backup job history.

```
SELECT SESSION_KEY,INPUT_TYPE,STATUS,START_TIME,END_TIME,ELAPSED_SECONDS/3600
hrs
FROM V$RMAN_BACKUP_JOB_DETAILS;

SESSION_KEY INPUT_TYPE    STATUS                  START_TIM END_TIME         HRS
----------- ------------- ----------------------- --------- --------- ----------
          8 DB FULL       FAILED                  27-MAR-12 27-MAR-12 1.64666666
         50 DB FULL       COMPLETED               28-MAR-12 28-MAR-12 .243055555
         69 DB FULL       COMPLETED               30-MAR-12 05-APR-12 147.176388
```

`SESSION_KEY` is the unique key for the RMAN session in which the backup job occurred.

Managing Backups

As part of a backup strategy, you must manage database backups. A related task is managing the record of those backups in the Oracle Recovery Manager (RMAN) repository. RMAN simplifies these tasks.

In a multitenant environment, you can manage backups for the whole multitenant container database (CDB) or for one or more pluggable databases (PDBs). The steps in this section are applicable to CDBs and PDBs, with minor modifications. To manage backups for the whole CDB, connect to the root and use the steps described in this section. To manage backups of a single PDB, connect to that PDB and use the steps described in this section. To manage backups of multiple PDBs using one command, connect to the root and use the `PLUGGABLE DATABASE` clause followed by a list of PDBs.

This section contains the following topics:

- Marking Backups as Available or Unavailable
- Deleting Obsolete Backups
- Monitoring Fast Recovery Area Space Usage

About Backup Management

An essential part of a backup and recovery strategy is managing backups after you create them. Backup management includes deleting obsolete backups and performing periodic checks to ensure that backups are available and usable.

A backup recorded in the Oracle Recovery Manager (RMAN) repository has one of the following status values:

- Available, meaning that the backup is still present on disk or tape, as recorded in the repository
- Expired, meaning that the backup no longer exists on disk or tape, but is still listed in the repository
- Unavailable, meaning that the backup is temporarily not available for data recovery operations (because, for example, it is stored on a tape that is stored offsite or on a disk that is currently not mounted)

Backups can also be obsolete. An obsolete backup is, based on the currently configured retention policy, no longer needed to satisfy data recovery goals.

Maintenance tasks that you can perform in RMAN include the following:

- Viewing details about your backups
- Cross-checking your repository, which means checking whether backups listed in the repository exist and are accessible, and marking as expired any backups not accessible at the time of the cross-check
- Deleting the record of expired backups from your RMAN repository
- Deleting obsolete backups from the repository and from the backup media
- Validating backups to ensure that a given backup is available and not corrupted

---

Note:

If a backup no longer exists, then immediately delete the backup record from the RMAN repository. Without an accurate record of available backups, you may discover that you no longer have complete backups of your database when you must perform a recovery.

Some tasks, such as periodic cross-checks of your backups, should be among the regularly scheduled components of your backup strategy.

If you use a fast recovery area for backup storage, then many maintenance activities are reduced or eliminated. The automatic storage space management mechanisms delete backups and other files as needed, thereby satisfying storage space demands for ongoing database operations without compromising the retention policy. However, you must monitor space usage in the fast recovery area to ensure that it is large enough to contain backups and other recovery-related files.

Cross-Checking Backups

Cross-checking a backup synchronizes the physical reality of backups with their logical records in the Oracle Recovery Manager (RMAN) repository. For example, if a backup on disk was deleted with an operating system command, then a cross-check detects this condition. After the cross-check, the RMAN repository correctly reflects the state of the backups.

Backups to disk are listed as available if they are still on disk in the location listed in the RMAN repository, and if they have no corruption in the file header. Backups on tape are listed as available if they are still on tape. The file headers on tape are not checked for corruption. Backups that are missing or corrupt are listed as expired.

To cross-check individual files:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Display a summary of the existing backups to determine which backup you want to cross-check.

   ```
   LIST BACKUP SUMMARY;
   ```

3. Identify the backup that you want to cross-check from the output of the previous `LIST` command.
4. Cross-check the identified file using the `CROSSCHECK` command.

   - To cross-check the backup set 1345:

     ```
     CROSSCHECK BACKUPSET 1345;

     using channel ORA_DISK_1
     crosschecked backup piece: found to be 'AVAILABLE'
     backup piece handle=/ade/b/191802369/oracle/work/orcva/RDBMS/backupset/
     2012_04_05/o1_mf_annnn_TAG20120405T075520_7qvdlrsl_.bkp RECID=1345
     STAMP=779788520
     Crosschecked 1 objects
     ```

   - To cross-check the data files 1 and 5:

     ```
     CROSSCHECK DATAFILECOPY 1,5;
     ```

To cross-check all files:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. To crosscheck all backup sets, use the following command:

   ```
   CROSSCHECK BACKUP;
   ```

Note:

Cross-checking all backups in the RMAN repository may take a long time, especially for tape backups. A cross-check of all files, unlike cross-checking individual files, is handled as a scheduled job.

See Also:

Oracle Database Backup and Recovery Reference for the syntax used to cross-check backups of multitenant container databases (CDBs) and pluggable databases (PDBs)

Deleting Expired Backups

Deleting expired backups removes from the Oracle Recovery Manager (RMAN) repository those backups that are listed as `EXPIRED`. Expired backups are those found to be inaccessible during a cross-check. No attempt is made to delete the files containing the backup from disk or tape; this action updates only the RMAN repository.

See Also:

Oracle Database Backup and Recovery Reference for the syntax used to delete expired backups of multitenant container databases (CDBs) and pluggable databases (PDBs)

To delete expired backups:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Optionally, cross-check backup sets by using the following command:

   ```
   CROSSCHECK BACKUPSET;
   ```

   Cross-checking backups before you delete expired backups provides RMAN with up-to-date information about which backups are expired.
3. Delete expired backups using the following command:

   ```
   DELETE EXPIRED BACKUP;
   ```

   The expired backup sets and image copies are deleted from the RMAN repository.

Marking Backups as Available or Unavailable

If one or more specific backups are unavailable because of a temporary condition, such as a disk drive that is temporarily offline or a tape stored offsite, then you can mark those backups as unavailable. Oracle Recovery Manager (RMAN) does not use unavailable backups to restore or recover data.

Note:

Backups stored in the fast recovery area cannot be marked as unavailable.

RMAN keeps the record of unavailable backups in the RMAN repository and does not delete backups listed as unavailable when you delete expired backups. If the unavailable backups become accessible again, then you can mark them as available.

See Also:

Oracle Database Backup and Recovery Reference for the syntax used for multitenant container databases (CDBs) and pluggable databases (PDBs)

To mark backups as available or unavailable:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Optionally, cross-check backup sets by using the following command:

   ```
   CROSSCHECK BACKUP;
   ```

   Cross-checking backups before you delete expired backups provides RMAN with up-to-date information about which backups are expired.
3. Display a summary of the available backups.

   ```
   LIST BACKUP SUMMARY;

   List of Backups
   ===============
   Key TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
   ------- -- -- - ----------- --------------- ------- ------- ---------- ---
   1 B A A DISK 24-FEB-07 1 1 NO TAG20070427T115348
   3 B A A DISK 24-MAR-07 1 1 NO TAG20070427T115452
   4 B F A DISK 24-APR-07 1 1 NO TAG20070427T115456
   ```
4. From the output of the `LIST` command, identify the backup that you want to make available or unavailable. Use the value displayed in the Key column to identify a backup set.
5. Change the status of the identified backup to unavailable by using the `CHANGE` command.

   The following command marks the backup set 4 unavailable:

   ```
   CHANGE BACKUPSET 4 UNAVAILABLE;

   changed backup piece available
   backup piece handle=/ade/b/191802369/oracle/work/orcva/RDBMS/backupset/
   2012_04_05/o1_mf_annnn_TAG20120405T075520_7qvdlrsl_.bkp RECID=23
   STAMP=779788520
   Changed 1 objects to AVAILABLE status
   ```

   To mark the backup set 4 available, use the following command:

   ```
   CHANGE BACKUPSET 4 AVAILABLE;
   ```

Deleting Obsolete Backups

This section explains how to delete obsolete backups, which are those no longer needed by the configured retention policy. If you use a fast recovery area as your only disk-based backup destination, then you never have to delete obsolete backups from disk. The fast recovery area keeps files as specified by the retention policy, and deletes them only when space is needed.

See Also:

Oracle Database Backup and Recovery Reference for the syntax used to delete obsolete backups of CDBs and PDBs

To delete obsolete backups:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Delete all obsolete backups, including backup sets and image copies, using the following command:

   ```
   DELETE OBSOLETE;
   ```

   RMAN displays the list of obsolete backups and asks for confirmation about deleting the listed backups.

Monitoring Fast Recovery Area Space Usage

It is important to monitor space usage in the fast recovery area to ensure that it is large enough to contain backups and other recovery-related files. Oracle Database provides two views to monitor fast recovery area space usage, V$RECOVERY_FILE_DEST and V$RECOVERY_AREA_USAGE.

Use the V$RECOVERY_FILE_DEST view to obtain the following information about the fast recovery area: total number of files, current location, disk quota, space in use, and space reclaimable by deleting files. The space details are in bytes. Querying V$RECOVERY_FILE_DEST produces the following output.

```
SELECT * FROM V$RECOVERY_FILE_DEST;

NAME         SPACE_LIMIT SPACE_USED SPACE_RECLAIMABLE NUMBER_OF_FILES
------------- ----------- ---------- ----------------- ---------------
/mydisk/rcva 5368709120  109240320  256000            28
```

The V$RECOVERY_AREA_USAGE view contains the percentage of disk quota used by different type of files, and the percentage of space that can be reclaimed by deleting files that are obsolete, redundant, or backed up to tape. Querying the V$RECOVER_AREA_USAGE view produces the following output.

```
SELECT * FROM V$RECOVERY_AREA_USAGE;

FILE_TYPE    PERCENT_SPACE_USED  PERCENT_SPACE_RECLAIMABLE  NUMBER_OF_FILES
------------ ------------------  -------------------------  ---------------
CONTROLFILE        0                   0                          0
ONLINELOG          2                   0                          22
ARCHIVELOG         4.05              2.01                         31
BACKUPPIECE        3.94              3.86                         8
IMAGECOPY         15.64              10.43                        66
FLASHBACKLOG       .08                0                          1
```

See Also:

"Configuring Your Database for Basic Backup and Recovery"

Performing Oracle Advised Recovery

The Oracle advised recovery feature uses Data Recovery Advisor, which is an Oracle Database feature that automatically diagnoses data failures, determines and presents appropriate repair options, and performs repairs if requested by the user. By providing a centralized tool for automated data repair, Data Recovery Advisor improves the manageability and reliability of an Oracle database.

Note:

Data Recovery Advisor can only be used to diagnose and repair failures in multitenant container databases (CDBs). It is not supported for pluggable databases (PDBs).

RMAN provides a command-line interface to the Data Recovery Advisor. You can use following RMAN commands to diagnose and repair data failures for the Oracle Database, including for Oracle RAC databases:

- `LIST FAILURE`

  Use this command to view problem statements for failures and the effect of these failures on database operations. Each failure is identified by a failure number.

- `ADVISE FAILURE`

  Use this command to view repair options, including both automated and manual repair options.

- `REPAIR FAILURE`

  Use this command to automatically repair failures listed by the most recent `ADVISE FAILURE` command.

See Also:

Oracle Database Backup and Recovery User's Guide for information about altering the status or priority of a failure by using the `CHANGE FAILURE` command

---

This section contains the following topics:

About Data Recovery Advisor

In the context of Data Recovery Advisor, a health check is a diagnostic procedure run by the Health Monitor to assess the state of the database or its components. Health checks are invoked reactively when an error occurs. You can also invoke checks manually.

A **failure** is a persistent data corruption detected by a health check. Failures are usually detected reactively. A database operation involving corrupted data results in an error, which automatically invokes a health check in the database. The check

searches the database for failures related to the error. If failures are diagnosed, then they are recorded in the Automatic Diagnostic Repository (ADR).

You can use Data Recovery Advisor to generate repair advice and repair failures only after failures have been detected by the database and stored in ADR. Data Recovery Advisor can report on and repair failures such as inaccessible files, physical and logical block corruptions, and I/O failures. Every failure has a failure priority: CRITICAL, HIGH, or LOW. Every failure also has a failure status of OPEN or CLOSED.

You can also use Data Recovery Advisor to view repair options. A **repair** is an action that fixes one or more failures. Examples of repairs include block media recovery, data file media recovery, and Oracle Flashback Database. Typically, Data Recovery Advisor presents both automated and manual repair options. If appropriate, you can choose an automated repair option in order to perform a repair. In this case, Data Recovery Advisor verifies the repair success, and closes the relevant repaired failures.

Using Data Recovery Advisor

The recovery process begins when you either suspect or discover a failure. You can discover failures in many ways, including error messages, alerts, trace files, and health checks. You can then use Data Recovery Advisor to gain information and advice about failures and repair them automatically.

This section describes a scenario in which you use Data Recovery Advisor to repair a corrupted block.

To use the Oracle advised recovery strategy to automatically repair failures:

1. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."

   In a multitenant environment, connect to the root as a common user with the SYSDBA or SYSBACKUP privilege.

2. List all the failures know to the Data Recovery Advisor by running the following command:

```
LIST FAILURE;

List of Database Failures
=========================
Failure ID Priority Status Time Detected Summary
---------- -------- ------ ------------- --------
142        HIGH     OPEN   23-APR-07     One or more non-system datafiles are
                                         missing
101        HIGH     OPEN   23-APR-07     Datafile 1: '/disk1/oradata/prod/
                                         system01.dbf' contains one or more
                                         corrupt blocks
```

   Wherever possible, RMAN consolidates failures while displaying the result of the `LIST FAILURE` command. For example, if a data file contains multiple block failures, the `LIST FAILURE` command consolidates and displays the repair options.

See Also:

Oracle Database Backup and Recovery User's Guide for information about using the `LIST FAILURE ... DETAIL` command to display failures individually

3. If you suspect that some failures that have not been automatically diagnosed by the database exist, then check for corrupt blocks and missing data files by using the following command:

   ```
   VALIDATE DATABASE;
   ```

   If a problem is detected during the validation, then RMAN triggers the execution of a failure assessment

4. Determine repair options, both automatic and manual, by using the following command:

   ```
   ADVISE FAILURE;
   ```

5. Fix the failures by using the following command:

   ```
   REPAIR FAILURE;
   ```

   Automated repairs are performed by the Data Recovery Advisor. In certain cases, such as when no backups exist for a lost control file, the only repair option possible is the manual option.

Performing User-Directed Recovery

User-Directed Recovery enables you to use flashback features and perform restore operations and recovery procedures. For example, you can do the following:

- Repair unwanted changes to database objects with the logical flashback features
- Rewind the entire database with Oracle Flashback Database
- Completely restore and recover the database
- Perform point-in-time recovery of the database or selected tablespaces
- Perform block media recovery of data files that have corrupted blocks

You can determine which parts of the database must be restored and recovered, including detecting situations such as corrupted database files before they affect database operations.

This section contains a few typical recovery examples so that you can become familiar with the performing whole database or object-level recovery using Oracle Recovery Manager (RMAN). Use the `RESTORE` and `RECOVER` commands to perform whole database or object-level recovery.

This section contains the following topics:

Rewinding a Table Using Oracle Flashback Table

Oracle Flashback Table enables you to rewind one or more tables back to their contents at a previous time without affecting other database objects. Thus, you can recover from logical data corruptions such as table rows added or deleted accidentally.

Unlike point-in-time recovery, the database remains available during the flashback operation.

For this example, you use Flashback Table on the `employees` table in the `hr` schema. Assume that an erroneous update shortly after October 23, 2005 at 15:30:00 has changed the `lastname` column for all employees to an empty string, and you must return the original `lastname` values to the table.

This section contains the following topics:

- Enabling Row Movement on a Table
- Performing a Flashback Table Operation

Enabling Row Movement on a Table

Before you can use Flashback Table, you must ensure that row movement is enabled on the table to be *flashed back*, or returned to a previous state. Row movement indicates that rowids will change after the flashback occurs. This restriction exists because if rowids before the flashback were stored by an application, then there is no guarantee that the rowids will correspond to the same rows after the flashback.

To enable row movement on a table:

1. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."
2. Enable row movement for all the objects that you want to rewind using Flashback Table.

   The following command enables row movement for the `hr.employees` table.

   ```
   ALTER TABLE hr.employees ENABLE ROW MOVEMENT;
   ```

   For this example, you must also enable row movement on the tables `hr.jobs` and `hr.departments`.

Performing a Flashback Table Operation

In this example, you rewind the `hr.employees` table and its dependent tables to a previous point in time.

To perform the Flashback Table operation:

1. Connect Oracle Recovery Manager (RMAN) to the target database as described in "Connecting to the Target Database Using RMAN."
2. Determine whether the table that you intend to flash back has dependencies on other tables.

   Use the following SQL query to determine the dependencies for the `hr.employees`:

   ```
   SELECT other.owner, other.table_name
       FROM sys.all_constraints this, sys.all_constraints other
       WHERE this.owner = 'HR'
         AND this.table_name = 'EMPLOYEES'
         AND this.r_owner = other.owner
         AND this.r_constraint_name = other.constraint_name
         AND this.constraint_type='R';

   OWNER                          TABLE_NAME
   ```

```
------------------------------ ------------------------------
HR                             EMPLOYEES
HR                             JOBS
HR                             DEPARTMENTS
```

3. Ensure that row movement is enabled for the table that you want to flash back and its dependent tables.

   In this example, row movement must be enabled for the tables `hr.employees`, `hr.jobs`, and `hr.departments` using the steps described in "Enabling Row Movement on a Table".

4. Identify the time, SCN, or restore point to which you want to return the table.

   In this example, we assume that the rows were accidentally inserted 5 minutes ago. Therefore, you must rollback to a timestamp that is 5 minutes before the current time.

   Note:

   If you do not know the time at which the unwanted changes occurred, you can use the Oracle Flashback Version Query to review all recent changes to the target table. Use of this feature is beyond the scope of this documentation.

5. Verify that enough undo data exists to rewind the table to the specified target.

   Use the following query to determine how long undo data is being retained:

```
SELECT NAME, VALUE/60 MINUTES_RETAINED
    FROM V$PARAMETER
    WHERE NAME = 'undo_retention';

NAME             MINUTES_RETAINED
---------------  ----------------
undo_retention               15
```

6. Use the `FLASHBACK TABLE` statement to perform a flashback operation for the required tables.

   The following SQL statements return the tables `hr.employees`, `hr.jobs`, and `hr.departments` to the specified time:

```
FLASHBACK TABLE hr.employees TO TIMESTAMP TO_TIMESTAMP('2012-03-27
09:30:00', 'YYYY-MM-DD HH:MI:SS');

FLASHBACK TABLE hr.jobs TO TIMESTAMP TO_TIMESTAMP('2012-03-27 09:30:00',
'YYYY-MM-DD HH:MI:SS');

FLASHBACK TABLE hr.departments TO TIMESTAMP TO_TIMESTAMP('2012-03-27
09:30:00', 'YYYY-MM-DD HH:MI:SS');
```

See Also:

- "Rewinding a Table Using Oracle Flashback Table"
- "Performing Oracle Advised Recovery"

Recovering a Dropped Table Using Oracle Flashback Drop

Oracle Flashback Drop enables you to reverse the effects of dropping (deleting) a table, returning the dropped table to the database along with dependent objects such as indexes and triggers. This feature stores dropped objects in a recycle bin, from which they can be retrieved until the recycle bin is purged, either explicitly or because space is needed.

As with Flashback Table, you can use Flashback Drop while the database is open. Also, you can perform the flashback without undoing changes in objects not affected by the Flashback Drop operation. Flashback Table is more convenient than forms of media recovery that require taking the database offline and restoring files from backup.

Note:

For a table to be recoverable using Flashback Drop, it must reside in a locally managed tablespace. Also, you cannot recover tables in the `SYSTEM` tablespaces with Flashback Drop, regardless of the tablespace type.

This section contains the following topics:

- Dropping a Table
- Retrieving a Dropped Table

Dropping a Table

For the purpose of learning about Flashback Drop, you will create a new table named `reg_hist` and then drop it. The database places the table in the recycle bin so that it can be retrieved with the Flashback Drop feature.

To create and then drop a table:

1. Connect SQL*Plus to the `hr` schema.
2. Create table `reg_hist` based on the existing `REGIONS` table in the `hr` schema by using the following command:

   ```
   CREATE TABLE reg_hist as SELECT * FROM REGIONS;
   ```

3. Drop the `reg_hist` table using the following command:

   ```
   DROP TABLE REG_HIST;
   ```

   Because Flashback is enabled for the database, the dropped table is stored in the recycle bin.

4. Display the tables in the `hr` schema.

   ```
   SELECT * FROM TAB;

   TNAME                          TABTYPE  CLUSTERID
   ------------------------------ ------- ----------
   BIN$ANbliLHaSiu02xI+zbvDvQ==$0  TABLE
   COUNTRIES                       TABLE
   DEPARTMENTS                     TABLE
   EMPLOYEES                       TABLE
   ```

```
EMP_DETAILS_VIEW              VIEW
JOBS                          TABLE
JOB_HISTORY                   TABLE
LOCATIONS                     TABLE
REGIONS                       TABLE

9 rows selected.
```

The first name displayed in the command output, beginning with 'BIN', is the table that you just dropped. Because Flashback Database is enabled, the deleted table is still in the recycle bin and is therefore displayed in the command output.

Retrieving a Dropped Table

This section assumes that you created and then dropped the `reg_hist` table, as described in "Dropping a Table." The following procedure retrieves `reg_hist` from the recycle bin.

To perform the Flashback Drop operation:

1. Connect SQL*Plus to the `hr` schema and obtain the name of the dropped table in the recycle bin.

```
SHOW RECYCLEBIN;

ORIGINAL NAME  RECYCLEBIN NAME                OBJECT TYPE   DROP TIME
-------------- ------------------------------ ------------  -----------
REG_HIST       BIN$ANbliLHaSiu02xI+zbvDvQ==$0  TABLE      2012-03-26:16:51:54
```

2. Retrieve the dropped table using the `FLASHBACK TABLE … TO BEFORE DROP` command.

   The following command performs a flashback of the `HR.REG_HIST` table.

```
FLASHBACK TABLE HR.REG_HIST TO BEFORE DROP;
```

> **Note:**
>
> When a table is retrieved from the recycle bin, all the dependent objects for the table that are in the recycle bin are retrieved with it. They cannot be retrieved separately.

3. If the retrieved table had referential constraints before it was placed in the recycle bin, then re-create them.

   This step must be performed manually because the recycle bin does not preserve referential constraints on a table.

Rewinding a Database Using Oracle Flashback Database

Unlike the other flashback features, Oracle Flashback Database operates at a physical level. When you use Flashback Database, your current data files revert to their contents at a previous time. The result is similar to database point-in-time recovery, but Flashback Database can be much faster because it does not require you to restore and recover data files. Also, Flashback Database requires limited application of redo data as compared to media recovery.

Flashback Database uses flashback logs to access previous versions of data blocks and also uses some data in the archived redo log files. To have the option of using

Flashback Database to repair your database, you must have configured the database to generate flashback logs as explained in "Configuring Recovery Settings."

Note:

You can use the Oracle Recovery Manager (RMAN) `FLASHBACK DATABASE` command to rewind the entire multitenant container database (CDB) only, not individual pluggable databases (PDBs).

To perform a Flashback Database operation:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Identify the desired SCN, restore point, or point in time to which the flashback database must be performed. This example rewinds the database to a specified point in time.

   See Also:

   Oracle Database Backup and Recovery User's Guide for information about how to determine the SCN and then flashback the database to this SCN

   ---

3. Shut down the database consistently, ensure that it is not opened by any instance, and then mount the database.

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ```

4. Flash back the database to the desired time.

   In this example, you need to flashback the entire database to the time specified in the `TIME` clause.

   ```
   FLASHBACK DATABASE to TIME "TO_DATE('03/20/12','MM/DD/YY')";
   ```

5. Open the database read-only and run some queries to verify the database contents.

   The following command opens the database in read-only mode:

   ```
   ALTER DATABASE OPEN READ ONLY;
   ```

6. After confirming that the state of the database is as expected, make the database available for updates by opening it with the `RESETLOGS` option.

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT;
   ALTER DATABASE OPEN RESETLOGS;
   ```

Restoring and Recovering the Database

This section demonstrates how to restore and recover the entire database using Oracle Recovery Manager (RMAN). This example assumes that you are restoring and recovering your database after the loss of one or more data files, but you still have a usable server parameter file and control file. You can also use RMAN to restore a lost server parameter file or control file.

To restore and recover the entire database:

1. Connect RMAN to the target database as described in "Connecting to the Target Database Using RMAN."
2. Ensure that the database is mounted, but not open.

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
```

3. Restore the database using the following command:

```
RESTORE DATABASE;
```

   The data files from the RMAN backup are restored to their default locations.

4. Recover the database using the `RECOVER` command.

```
RECOVER DATABASE;
```

5. Open the database using the following command:

```
ALTER DATABASE OPEN;
```

See Also:

- *Oracle Database Backup and Recovery User's Guide* for information about recovering multitenant container databases (CDBs) and pluggable databases (PDBs)
- *Oracle Database Backup and Recovery User's Guide* for instructions for recovering a database when the control file and server parameter file are available only on backup
- *Oracle Database Backup and Recovery User's Guide* for information about restoring data files to a location that is different from the default location

Backup and Recovery: Oracle By Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this section, and includes annotated screenshots. To view the Performing Backup and Recovery OBE, enter the following URL in your web browser:

`https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:24:P24_CONTENT_ID,P24_PREV_PAGE:6289,1`

10

Monitoring and Tuning the Database

Monitoring the performance of a database and ensuring that it performs optimally is an important task for a database administrator. This chapter discusses the features and functions included in Oracle Database that make it easy to monitor database health, identify performance problems, and implement any corrective actions.

This chapter contains the following topics:

- Proactive Database Monitoring
- Diagnosing Performance Problems Using ADDM
- Using Advisors to Optimize Database Performance
- Monitoring and Tuning: Oracle by Example Series

Proactive Database Monitoring

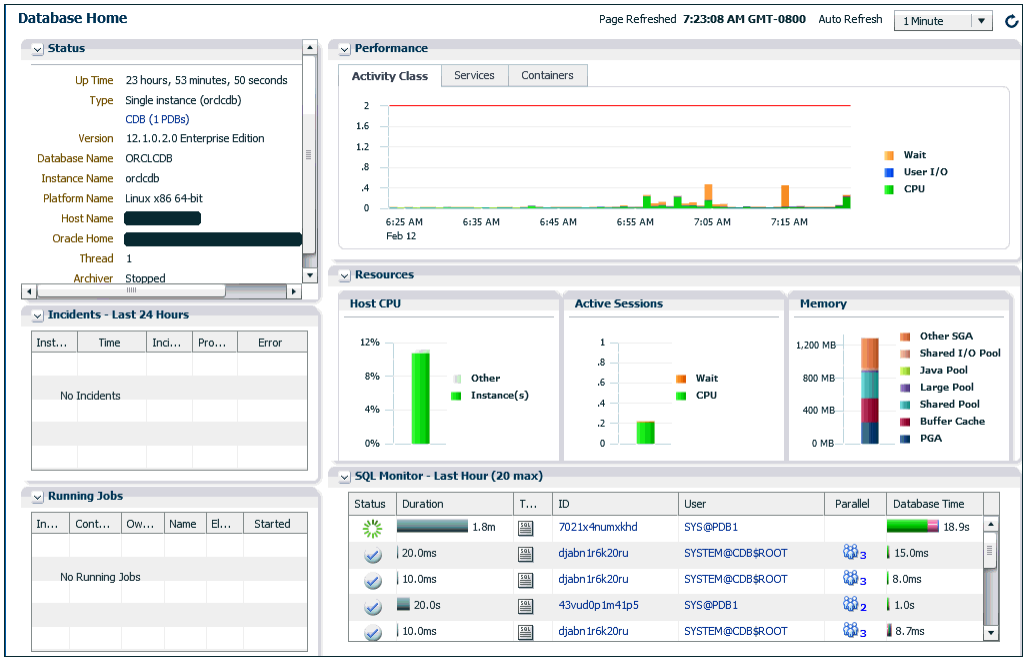
Oracle Database makes it easy to monitor the health and performance of your database. It monitors the vital signs (or metrics) related to database health and performance, analyzes the workload running against the database, and automatically identifies any issues that need your attention as an administrator. Any incidents (critical errors in the database) are reported on the Database Home page in EM Express.

This section discusses the following topics:

- Monitoring General Database State and Workload
- Monitoring Performance Using the Performance Hub
- Performance Self-Diagnostics: Automatic Database Diagnostic Monitor

Monitoring General Database State and Workload

The Database Home page enables you to monitor the state and workload of your database. It provides a central place for general database state information and is updated periodically.



To monitor the general database state and workload:

1. Go to the Database Home page.

   See "Accessing the Database Home Page".

2. (Optional) Click the Refresh icon to the right of the selected refresh interval for the **Auto Refresh** list to update the information displayed.

   The time that the Database Home page was last collected from the database appears near the top right corner of the page.

   By default, the Database Home page automatically refreshes every 60 seconds. You can prevent automatic refresh by selecting **Off** in the **Auto Refresh** list at the top right-hand corner of the page. You must then click the Refresh icon to view the latest information.

3. Get a quick overview of the database state in the Status section, which includes the following information:

   - Up Time

     Information about how long the database has been up

   - Type

     The database type. The type can be a single instance database (CDB or non-CDB) or an Oracle RAC database (or cluster database).

     If the database type is a CDB, the next line will identify the database as a CDB and specify the number of PDBs in the CDB. The **CDB (n PDBs)** line is a link to the Containers page, which shows a list of containers in the CDB (not including PDB$SEED), as well as status, performance, and resource information about the containers.

   - Version

     The database version number

- Database name

  The database name

- Instance name

  The name of the database instance

- Platform name

  The platform on which the database is running

- Host name

  The name of the host system on which the database is running

- Thread

  The redo log threads for the database

- Archiver

  The status of the archiver process

4. View active session information in the Performance section. The Performance section shows trend information for the past hour.

   The Activity Class chart shows the average number of database sessions active for the past hour. The chart shows the type of activity for each session (on CPU, waiting for I/O, or waiting for another resource).

   The Services chart shows the average number of database sessions active for the past hour for database services.

   For Oracle RAC, the Activity Class chart shows activity aggregated across all instances in the cluster. Also, an Instances chart appears for Oracle RAC that shows Average Active Sessions per instance.

5. View resource utilization for the latest data point (the last minute) in the Resources section. The Resources section includes the following information:

   - Host CPU chart

     This chart shows the percentage of CPU time used by the database instance and other processes during the last minute. Place your cursor over the instance data to see the percentage of CPU used by foreground and background instance processes.

     If other processes are taking up most of your CPU time, then this indicates that some other application running on the database host computer could be causing performance problems.

   - Active Sessions chart

     This chart shows the average number of active sessions during the last minute, broken out by wait, user I/O, and CPU.

     - Waits

       This is the value for all wait classes combined, excluding user I/O and idle wait events. **Wait classes** are groupings of wait events based on the type of wait.

Go to the Performance Hub and click the Activity tab to view more information about waits.

- User I/O

  This is the average number of active sessions waiting for user I/O. User I/O means that the workload originating from the user causes the database to read data from disk or write data to disk.

  Go to the Performance Hub and click the Summary tab to view more information about I/O.

- CPU

  This is the average active sessions using CPU.

  Go to the Performance Hub and click the Summary tab to view more information about CPU usage.

- Memory (GB)

  This chart shows the current memory utilization (as of the latest refresh time) broken out by the database shared pool, java pool, buffer cache, PGA, and other SGA components.

- Data Storage (GB)

  This chart shows the current space usage (as of the latest refresh time) broken out by user data, database log files, undo tablespaces, and temporary, SYSAUX, and SYSTEM tablespaces.

6. View SQL activity in the SQL Monitor section:

   The table in this section displays information about monitored SQL statement executions. If there is a green spinning icon in the Status column, then the monitored statement is still running. If there is a check mark in the Status column, then the statement has completed its execution.

   SQL statements are monitored only if they have consumed at least 5 seconds of CPU or I/O time.

   For each SQL statement, the table provides information in the Status, Duration, SQL ID, Session ID, Parallel, Database Time, and SQL Text columns.

   Click a SQL ID to display the SQL Details page with more information about that SQL statement.

7. The Incidents - Last 24 Hours section displays a table that provides information about database incidents that have occurred in the past 24 hours. The table has the Instance, Time, Incident, Problem, and Error columns.

   An incident is an occurrence of a critical error in the database. Each incident in the Incidents - Last 24 Hours table is recorded in the Automatic Diagnostic Repository (ADR), a directory structure located outside the database, which is available for problem diagnosis even when the database is down. You can investigate critical errors using the ADR Command Interpreter (ADRCI) utility.

See Also:

- *Oracle Database SQL Tuning Guide* for more information about the Enterprise Manager pages for monitored SQL executions
- "About the SQL Tuning Advisor" for an introduction to SQL Tuning Advisor
- "About the Automatic SQL Tuning Advisor" for an introduction to Automatic SQL Tuning Advisor
- "Running the SQL Tuning Advisor" for an example of tuning a SQL statement using SQL Tuning Advisor
- *Oracle Database Administrator's Guide* for information about investigating problems using ADRCI.

8. The Running Jobs section displays a table that shows database jobs that are currently running. The table has the Instance, Owner, Name, Elapsed, and Started columns.

Monitoring Performance Using the Performance Hub

The Performance Hub allows you to view all the performance data available for a specified time period. Once a time period is selected, the performance information is collected and presented based on performance subject areas.

This section includes the following topic:

- Specifying the Time Period for Which to Display Statistics

When real-time data is selected, more granular data is presented (because data points are available every minute).

When historical data is selected, more detailed data (broken down by different metrics) is presented, but the data points are averaged out to the Automatic Workload Repository (AWR) interval (usually an hour).

Different tabs are available in the Performance Hub, depending on whether real-time or historical data is selected for the time period.

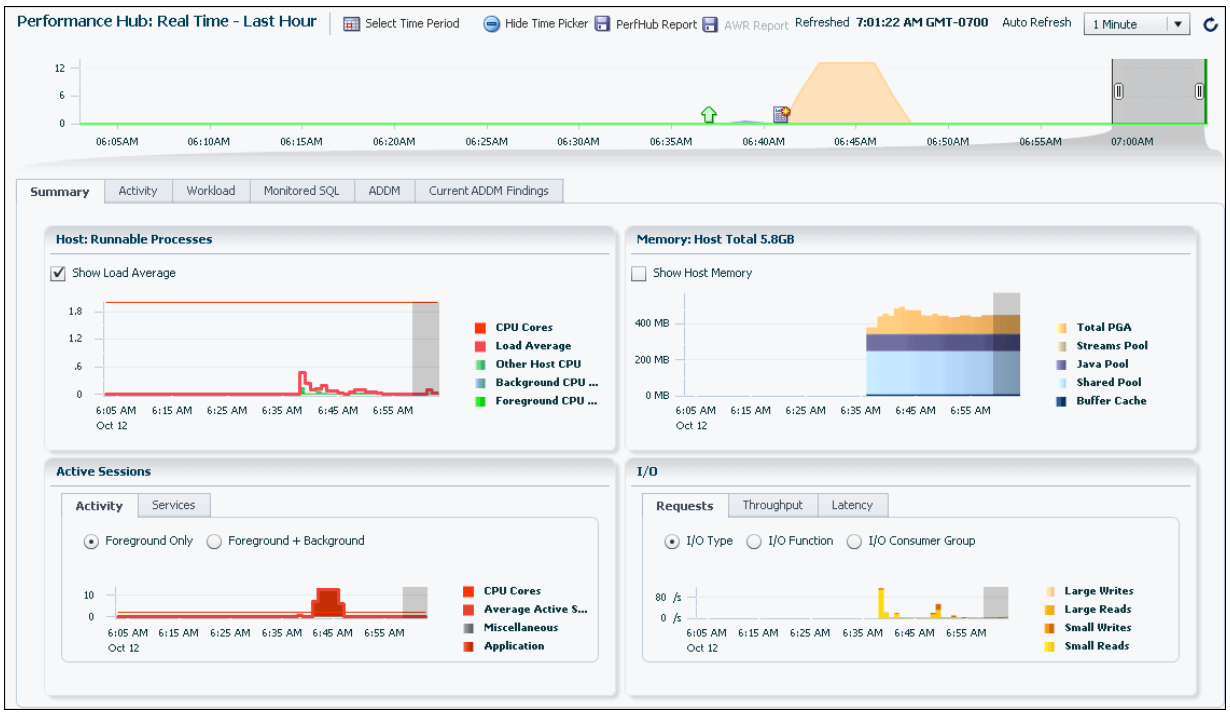
The following table describes the Performance Hub tabs, and indicates whether the tab is available when real-time data is selected or historical data is selected, or both.

Performance Hub Tab Name	Description	Available When
Summary	The Summary tab provides an overall view of the performance of the system for the specified time period. When real-time data for the last hour is displayed in the Performance Hub page, this tab shows a summary of running processes, memory allocation, database activity by category, and I/O data during the last hour. When historical data is displayed in the Performance Hub page, this tab shows a summary of average active session waits by	Real-time data or historical data is selected in the Select Time Period field

Performance Hub Tab Name	Description	Available When
	category, load profile per second, active session activity, host CPU usage by the database instance and other processes, I/O read and write requests per second, and memory usage during the selected time period.	
RAC	The RAC tab appears only when EM Express is being used with an Oracle RAC database (or cluster database). When real-time data is selected, this tab shows global cache activity information and a breakdown of activity (average active sessions) and resource usage (CPU, I/O, memory) per instance. When historical data is selected, this tab shows global cache activity information and a breakdown of activity (average active sessions) and resource usage (CPU, I/O, memory) per instance during the selected time period.	Real-time data or historical data is selected in the Select Time Period field
Activity	The Activity tab shows Active Session History (ASH) analytics. It allows detailed drilldown into average active sessions for ASH over the selected time period. This tab enables you to select an average active sessions dimension and view the top activity for that dimension for a selected time period. For example, you can view the SQL statements and user sessions that had the top average active sessions activity for the selected time period.	Real-time data or historical data is selected in the Select Time Period field
Workload	The workload profile charts show the pattern of user calls, parse calls, Redo Size and SQL*Net over the last 60 minutes in real-Time mode. The Sessions chart show the logon rate, current logons and open cursors. Clicking a SQL_ID displays the SQL Details page with more information about that SQL statement.	Real-time data or historical data is selected in the Select Time Period field
Monitored SQL	This tab enables you to view information about monitored SQL statements that were executing or that completed during the selected time period. The table displays information about monitored SQL statement executions. If there is a green spinning icon in the Status column, then the monitored statement did not complete during the selected time period. A red cross indicates that the SQL did not complete either due to an error or due to the session getting terminated. If there is a check	Real-time data or historical data is selected in the Select Time Period field

Performance Hub Tab Name	Description	Available When
	mark in the Status column, then the statement completed its execution during the selected time period. SQL statements are monitored only if they have consumed at least 5 seconds of CPU or I/O time. You can view information such as the status of a statement, its duration, its type (SQL, PL/SQL, or DBOP), its SQL ID, its SQL plan hash, the user who issued it, whether it executed as a serial or parallel statement, the time the database spent performing CPU activity, I/O, or other activity for the statement, the read and write requests and bytes associated with the statement, and the start and end time for the statement. Click a SQL ID to display the SQL Details page with more information about that SQL statement.	
ADDM	The ADDM tab enables you to view performance findings and recommendations that have been found by Automatic Database Diagnostics Monitor (ADDM) for tasks performed in the database during the selected time period. See "Performance Self-Diagnostics: Automatic Database Diagnostic Monitor " for more information about ADDM features.	Real-time data or historical data is selected in the Select Time Period field
Current ADDM Findings	The Current ADDM Findings tab enables you to view Real-time ADDM findings for the past five minutes. See "Performance Self-Diagnostics: Automatic Database Diagnostic Monitor " for more information about ADDM features.	Real-time data is selected in the Select Time Period field
Database Time	The Database Time tab enables you to view wait events by category for various metrics, and to view time statistics for various metrics for the selected time period.	Historical data is selected in the Select Time Period field
Resources	The Resources tab enables you to view operating system resource usage statistics, I/O resource usage statistics, and memory usage statistics for the selected time period.	Historical data is selected in the Select Time Period field
System Statistics	The System Statistics tab enables you to view database statistics by value, per transaction, or per second for the selected time period.	Historical data is selected in the Select Time Period field

The following figure shows the Performance Hub when Real Time - Last Hour data is selected.



The following figure shows the Performance Hub when historical data is selected.

To view Performance Hub data:

1. At the top of the Database Home page, from the **Performance** menu, select **Performance Hub**.

   The Performance Hub page appears, with the Summary tab displayed. By default, real time data for the last hour appears in the Performance Hub.

   You can select a different time period in the **Select Time Period** field if you would like to view historical data in the Performance Hub instead of real-time data.

   In the figure above, **Historical - All** is selected in the **Select Time Period** field.

2. The time picker appears below the **Select Time Period** field.

   The shaded block area in the time picker identifies the period of time for which performance statistics are currently being displayed in the Performance Hub. This is a subset of the period of time you selected in the Select Time Period field.

   When historical data is displayed in the Performance Hub, you can increase or decrease the size of the shaded block area by clicking and dragging the user control on either end of the shaded block area.

   The shaded block area is the time period for which statistics are displayed on all of the Performance Hub tabs, not just on the currently selected tab.

3. Click any of the tabs that appear in the Performance Hub to view the performance data on the tab.

4. Click the **PerfHub Report** button to generate a Performance Hub active report, which will include the contents of the Performance Hub tabs in an HTML file. After you click **PerfHub Report**, you are prompted to choose one of these report types for the Performance Hub active report:

   - Basic: The basic information for all the tabs is saved in the report.
   - Typical: All the information for the basic report is saved. Also, the SQL Monitor information for the top SQL in the Monitored SQL tab is saved, and ADDM reports are saved.
   - All: All the information for the basic report is saved. Also, the SQL Monitor information for all of the SQL in the Monitored SQL tab is saved (not just for the top SQL), and all the detailed reports in all the tabs are saved.

   You are then prompted for a location and file name for the active report, and the report is generated in that file and location. You use a web browser to view the report and navigate the Performance Hub tabs in the report.

5. When historical data is selected in the Performance Hub, you can click the **AWR Report** button to generate an AWR report for the selected time period.

   You are prompted for a location and file name for the AWR report, and the report is generated in that file and location. You use a web browser to view the report.

See Also:

Oracle Database Performance Tuning Guide for more information about AWR

Specifying the Time Period for Which to Display Statistics

In the Real-Time: Last Hour mode, the data in the Performance Hub is sourced from Active Session History (ASH). The ASH data is written to disk when the ASH buffer is filled up or after 1 hour, and is stored as part of the AWR framework.

By default, AWR has a retention period of 8 days. When you view historical data in the Performance Hub, you are viewing statistics collected as part of the hourly snapshots in AWR.

You use the Select Time Period field in the Performance Hub to determine the time periods for which statistics are available for viewing. Because Oracle Database statistics are stored in memory for one hour, the Real Time - Last Hour option always appears in the Select Time Period list.

The historical data options that are available in the Select Time Period list change, depending on the time period for which data is available in AWR, as shown in the following table:

Time Period for Which AWR Data is Available	Historical Options in the Select Time Period List
Less than 24 hours	Historical - All
More than 24 hours, but less than 7 days	Historical - Day Historical - All[1] Historical - Custom
7 days	Historical - Day Historical - Week Historical - Custom
8 days or more[2]	Historical - Day Historical - Week Historical - Custom

1 This option is available only when less than one day's data or less than one week's data is available in AWR. Database statistics that are stored in memory are flushed to AWR after one hour.

2 The default AWR retention period is 8 days, so you must change the default AWR retention period to store more than a week of data in AWR

After you choose a historical option from the Select Time Period field, you use the time picker to specify the time period for which data is displayed in the Performance Hub tabs.

The following table describes the data displayed and available for selection in the time picker when different values are selected in the Select Time Period field for the Performance Hub:

Selected Time Period	Time Picker	Description
Real Time - Last Hour	Displays statistics for the past hour from memory	Data is displayed in 5 minute blocks in the time picker. Use the time picker to select from 1 minute to 60 minutes of data to display in the Performance Hub.
Historical - Day	Displays statistics from an hour up to 24 hours from AWR	Data is displayed in 1 hour blocks in the time picker. Use the time picker to select from 1

Selected Time Period	Time Picker	Description
		hour to 24 hours of data to display in the Performance Hub.
Historical - All	Displays statistics for the length of time for which AWR statistics exist	Appears only when less than one day's data or less than one week's data is available in AWR. The Historical - All option is available only when there is not enough AWR data to provide the Historical - Day option or the Historical - Week option.
Historical - Week	Displays statistics from a day up to 7 days from AWR	Data is displayed in 1-day blocks in the time picker. Use the time picker to select from 1 day to 7 days of data to display in the Performance Hub. When Historical - Week is selected, the current week of AWR data appears in the time picker by default. To view AWR data from the previous week in the time picker, use the < button in the time picker.
Historical - Custom	Displays AWR statistics for the length of time you select in the Select Time Period dialog box after choosing the Historical - Custom option	Use the time picker to select the time period for which you want to display statistics in the Performance Hub.

See Also:

- *Oracle Database Concepts* for more information about ASH
- *Oracle Database Performance Tuning Guide* for more information about changing the default AWR retention period

Performance Self-Diagnostics: Automatic Database Diagnostic Monitor

Oracle Database includes a self-diagnostic engine called Automatic Database Diagnostic Monitor (ADDM). ADDM makes it possible for Oracle Database to diagnose its own performance and determine how identified problems can be resolved.

To facilitate automatic performance diagnosis using ADDM, Oracle Database periodically collects snapshots of the database state and workload. **Snapshots** are sets of historical data for specific time periods that are used for performance comparisons by ADDM. Snapshots provide a statistical summary of the state of the system at a point in time. These snapshots are stored in Automatic Workload Repository (AWR), residing in the SYSAUX tablespace. The snapshots are stored in this repository for a set time (8 days by default) before they are purged to make room for new snapshots.

ADDM analyzes data to determine the major problems in the system, and may recommend solutions and quantify expected benefits. ADDM analysis results are represented as a set of **findings**.

EM Express provides three types of ADDM findings.

ADDM

ADDM performs its analysis on data that has been captured and stored in AWR. For ADDM, the default collection interval for a snapshot is one hour.

Generally, ADDM is used for identifying systemwide systemic problems. It calls attention to performance problems that include:

- Resource contention (bottlenecks), such as when your database is using large amounts of CPU time or memory due to high load SQL statements
- Poor connection management, such as when your application is making too many logins to the database
- Lock contention in a multiuser environment, such as when one user process acquires a lock to safely update data in a table, causing other user processes that must acquire locks against the same table to wait, resulting in a slower database performance

Real-Time ADDM

Real-Time ADDM automatically monitors the database in real time.

Real-Time ADDM proactively detects and diagnoses transient high impact problems such as these before they threaten application performance:

- High CPU
- I/O spikes
- Memory
- Interconnect issues
- Hangs and deadlocks

When Real-Time ADDM detects a possible performance problem, it triggers data collection. The data is saved in the report repository (part of AWR). When you view a Real-Time ADDM report from EM Express, an analysis is performed, and findings and recommendations are made. Because Real-Time ADDM reports are stored in AWR, they can help you identify recurrences of a problem over time.

Current ADDM Findings

If there are Real-Time ADDM findings for the past 5 minutes, these are listed on the Current ADDM Findings tab in the Performance Hub when both of the following are true:

- Real-Time Data - Last Hour is selected in the Select Time Period field in the Performance Hub
- The selected period in the time picker in the Performance Hub is "now" (the shaded blocks area in the time picker is at the far right in the Performance Hub)

Table 1 provides a summary of the ADDM features available in EM Express.

Table 1 ADDM Features in EM Express

Feature	New?	Description	Analysis Period	To View Analysis Findings
ADDM	No	This is the traditional ADDM that has existed in previous database releases. ADDM Tasks are presented on the ADDM tab in the Performance Hub.	The AWR interval, which is 1 hour, by default	In the ADDM Tasks table on the ADDM tab, click a **Task Name**.
Real-Time ADDM	Yes	Proactively detects and diagnoses transient high impact problems in real time. Real-Time ADDM Reports are presented on the ADDM tab in the Performance Hub.	Real time	In the Real-Time ADDM Reports table on the ADDM tab, select a report and click **View Performance Report**.
Current ADDM Findings	Yes	Triggers Real-Time ADDM analysis and findings for the past 5 minutes. Current ADDM Findings are presented on the Current ADDM Findings tab in the Performance Hub.	Last 5 minutes	In the Current ADDM Findings table, select a finding and click **View Finding**.

See Also:

- *Oracle Database 2 Day + Performance Tuning Guide* for more information about automatic database performance monitoring
- "Diagnosing Performance Problems Using ADDM"

Diagnosing Performance Problems Using ADDM

At times, database performance problems arise that require your diagnosis and correction. Usually, these problems are brought to your attention by ADDM, which analyzes data for different time periods.

This section contains the following topics:

- Viewing a Summary of ADDM Performance Findings
- Responding to ADDM Performance Findings
- Viewing a Summary of Real-Time ADDM Findings

- Responding to Real-Time ADDM Findings
- Viewing a Summary of Current ADDM Findings
- Responding to Current ADDM Findings

See Also:

Oracle Database 2 Day + Performance Tuning Guide

Viewing a Summary of ADDM Performance Findings

ADDM analysis results consist of a description of each finding and a recommended action. You can view a summary of findings and their impacts on the system.

To view a summary of ADDM performance findings:

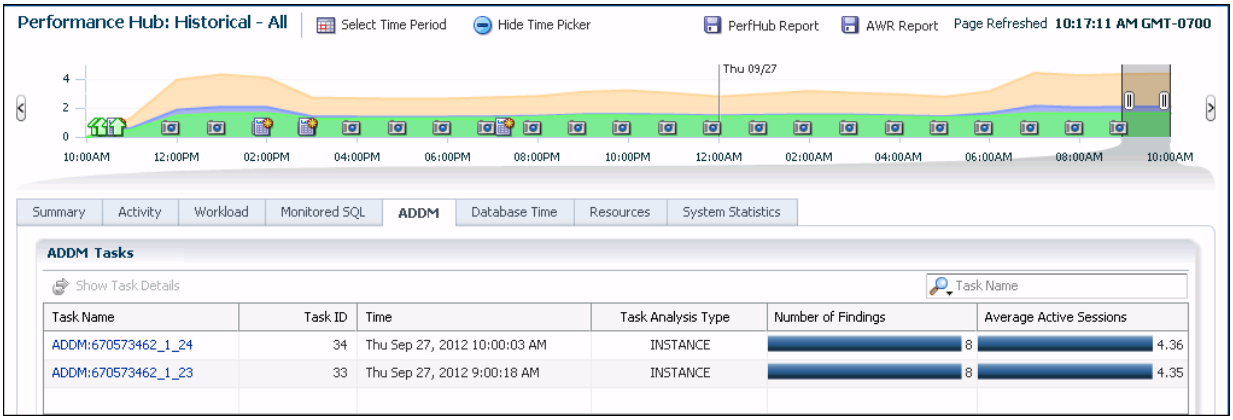
1. In EM Express, from the **Performance** menu, select **Performance Hub**.

   The Performance Hub page appears in the list of tabs on the Performance Hub page.

2. In the **Select Time Period** field, select one of the time period values to view performance data for that time period. The title of the page changes to indicate the selected time period.

3. Click the ADDM tab.

   The ADDM tasks for the selected time period appear in the ADDM Tasks table on the ADDM tab.



4. Click the link in the **Task Name** column for a ADDM task to view more information about that task.

See Also:

"Performance Self-Diagnostics: Automatic Database Diagnostic Monitor "

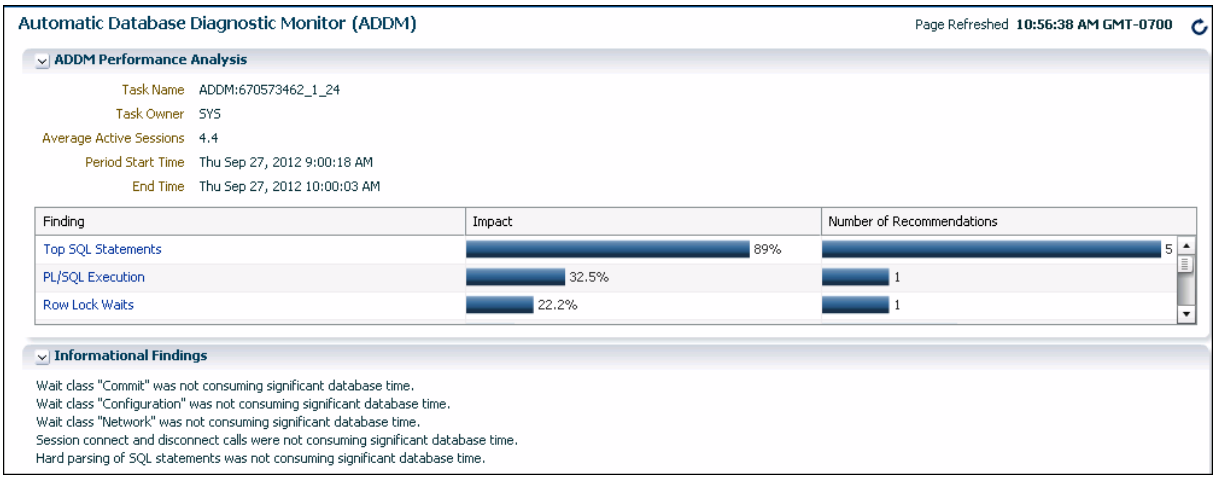
Responding to ADDM Performance Findings

You can act upon the recommendations that accompany ADDM performance findings.

To respond to ADDM performance findings:

1. In the ADDM Tasks table on the ADDM tab of the Performance Hub page, click the link in the **Task Name** column to view the performance findings for that task.

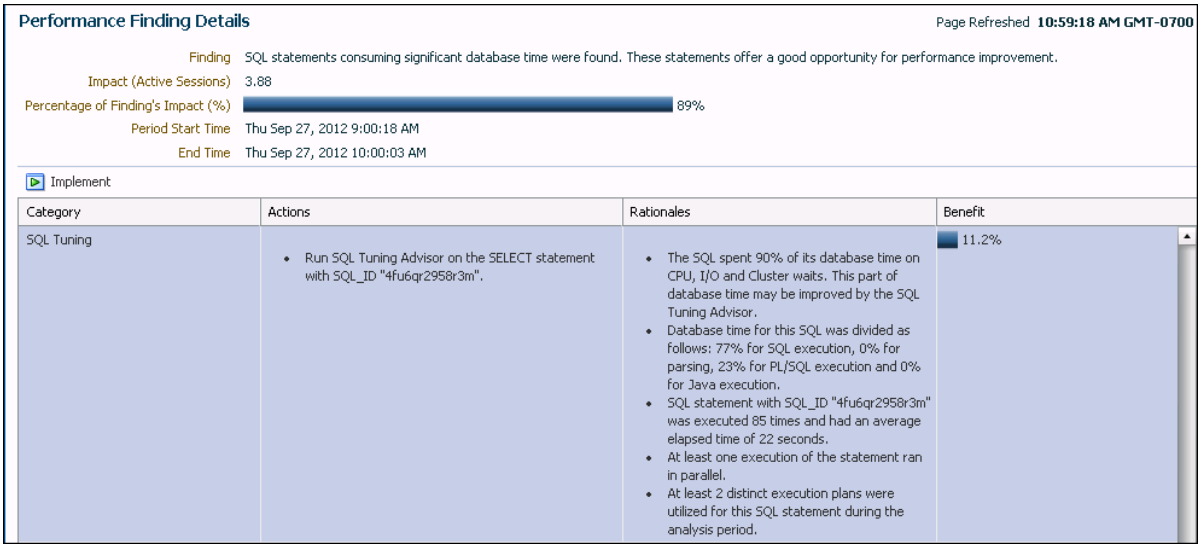
   The Automatic Database Diagnostic Monitor (ADDM) page appears.



   The performance findings for the ADDM task and the performance impact of each finding (%) are listed in the Findings table.

2. Click the link in the **Finding** column for a particular finding to view the finding and recommendations (if any) associated with it.

   The Performance Finding Details page appears.



   The performance findings table lists the performance findings for the ADDM task. For each finding, the table shows the finding category, recommended action,

rationale for the recommendations, and the expected benefit of implementing the recommendation.

When you select a finding that has a recommendation that you can implement using ADDM, the Implement button becomes available for that finding.

3. For a finding that has a recommendation that you want to implement using ADDM, select the finding and click **Implement**.

   For this example, the Schedule SQL Tuning Advisor page appears, on which you are prompted for the information necessary to run the SQL Tuning Advisor on the selected SQL statement.

See Also:

"Performance Self-Diagnostics: Automatic Database Diagnostic Monitor "

Viewing a Summary of Real-Time ADDM Findings

Real-Time ADDM results consist of a description of each finding and recommended actions for some findings at the point in time when the Real-Time ADDM report was generated. You can view a summary of findings and their impacts on the system.

To view a summary of Real-Time ADDM performance findings:

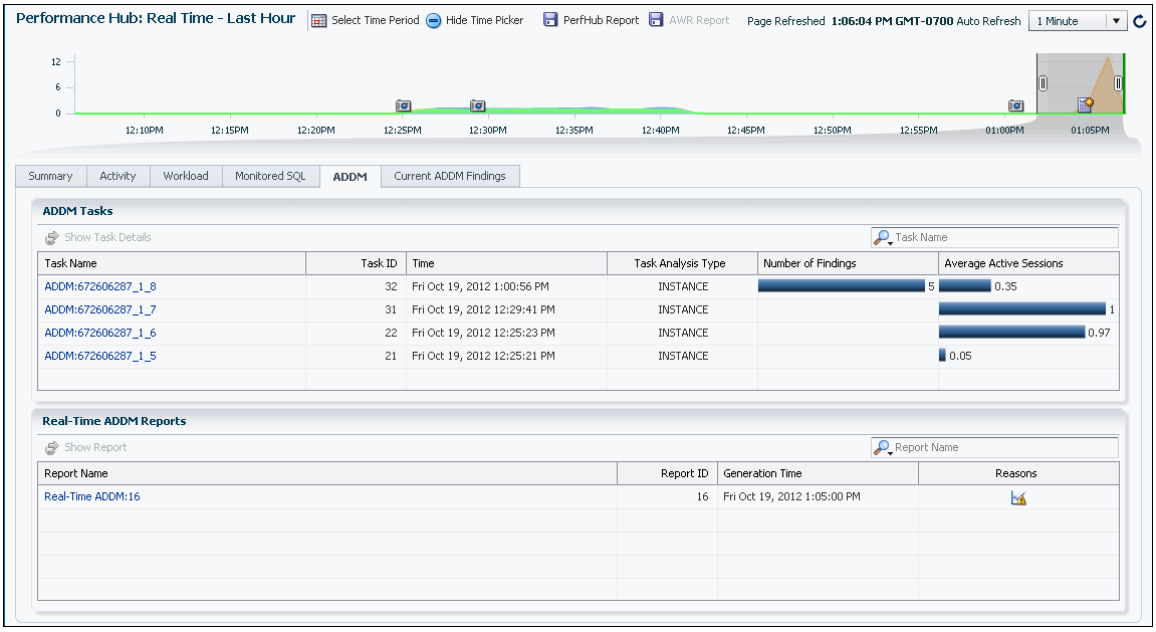
1. In EM Express, from the **Performance** menu, select **Performance Hub**.

   The Performance Hub page appears in the list of tabs on the Performance Hub page.

2. In the **Select Time Period** field, select one of the time period values to view performance data for that time period. The title of the page changes to indicate the selected time period.

3. Click the ADDM tab.

   If there are Real-Time ADDM reports for the selected time period, the reports appear in the Real-Time ADDM Reports table at the bottom of the ADDM tab.



4. To view a particular report in the table, click the link for the report in the **Report Name** field, or select the table row and then click **Show Performance Report**.

See Also:

"Performance Self-Diagnostics: Automatic Database Diagnostic Monitor "

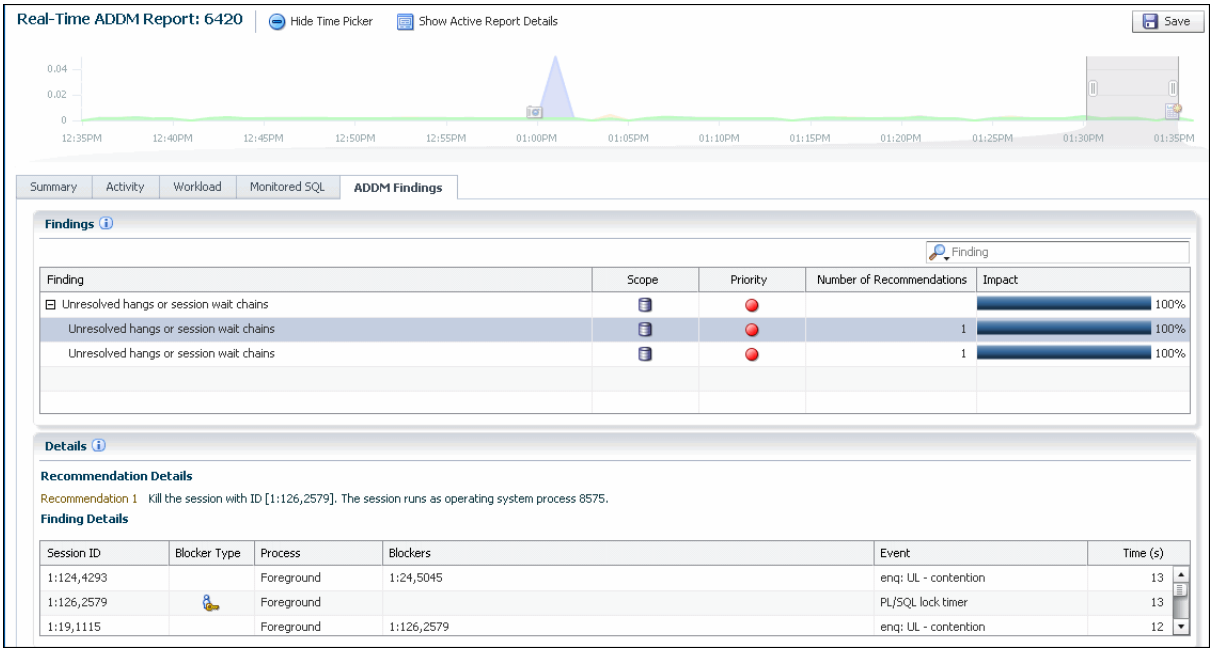
Responding to Real-Time ADDM Findings

You can view the findings in a Real-Time ADDM report.

To respond to Real-Time ADDM performance findings:

1. To view a particular report in the Real-Time ADDM Reports table, click the link for the report in the **Report Name** field, or select the table row and then click **Show Performance Report**.

   The Real-Time ADDM Report page appears.



2. In the Findings table, the findings for the report, and the impact of each finding are displayed. When you select a finding in the Findings table, the recommendation details and finding details for that finding are displayed in the Recommendation Details and Finding Details sections below the Findings table.

See Also:

"Performance Self-Diagnostics: Automatic Database Diagnostic Monitor "

Viewing a Summary of Current ADDM Findings

Current ADDM Findings analysis results consist of a description of each current finding and a recommended action. You can view a summary of current ADDM findings and their impacts on the system.

To view Current ADDM Findings:

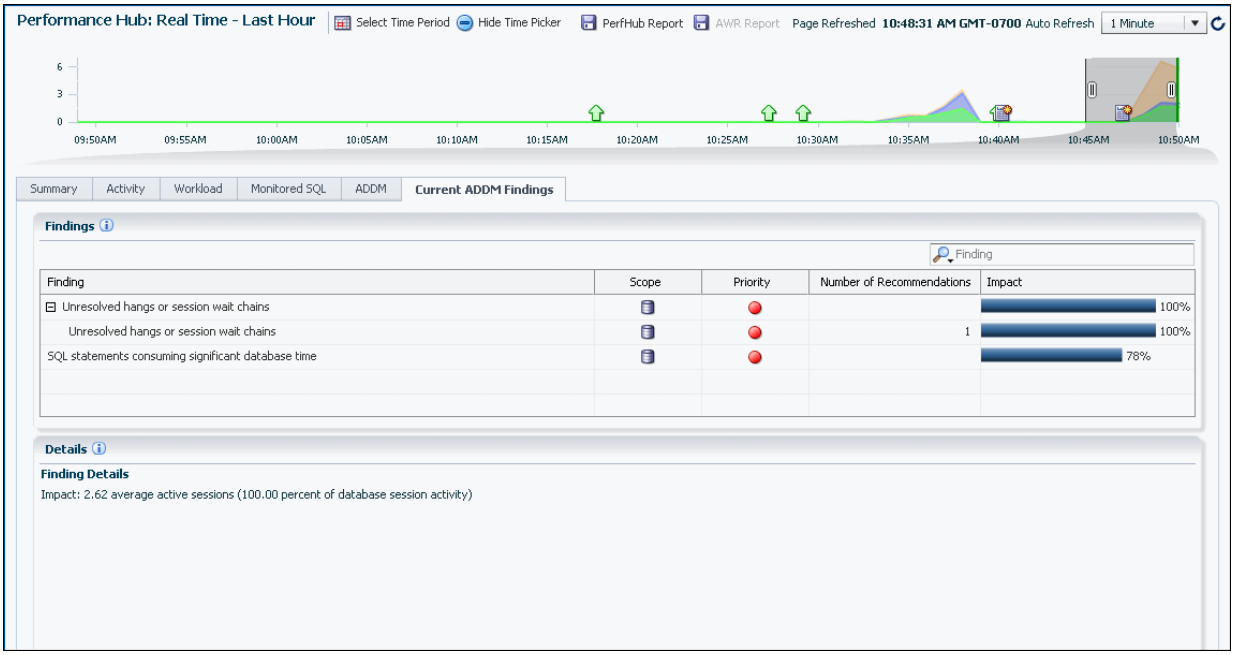
1. In EM Express, from the **Performance** menu, select **Performance Hub**.

   The Performance Hub page appears in the list of tabs on the Performance Hub page.

2. In the **Select Time Period** field, select **Real-Time: Last Hour**. Current ADDM Findings are only available for the past 5 minutes of real time data.

3. Click the Current ADDM Findings tab.

   If there are current ADDM findings for the past 5 minutes in real time, those findings appear in the Findings section of the Current ADDM Findings tab.



See Also:

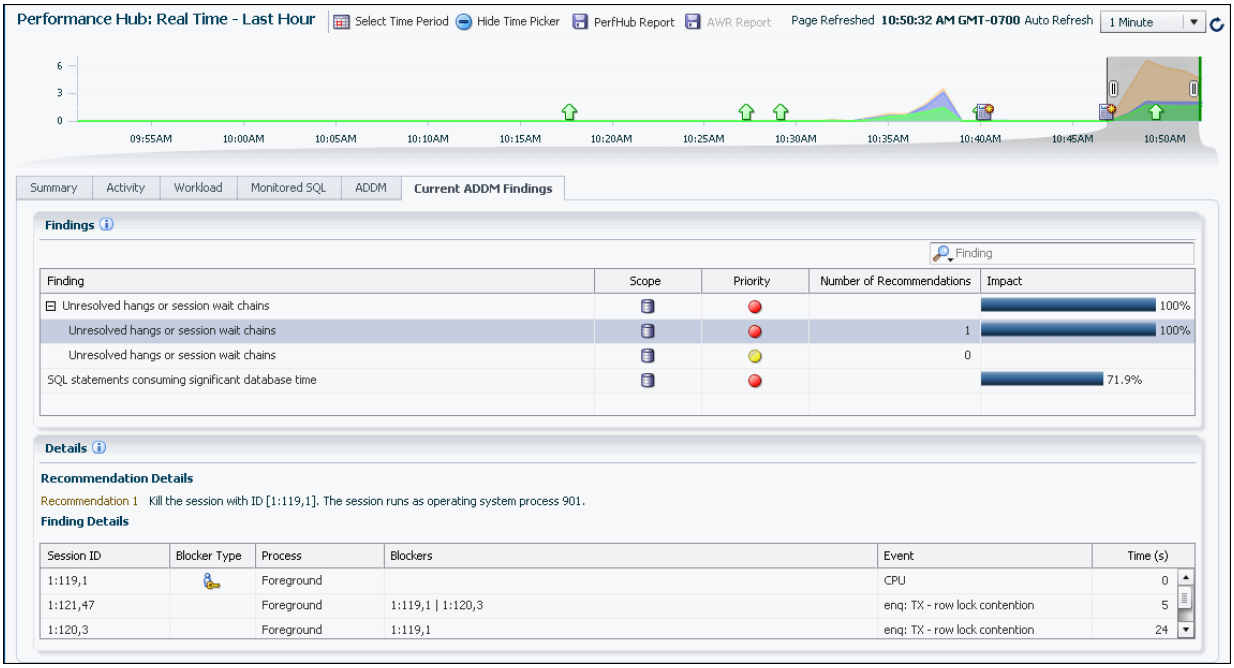
"Performance Self-Diagnostics: Automatic Database Diagnostic Monitor "

Responding to Current ADDM Findings

You can respond to the current findings on the Current ADDM Findings tab.

To respond to current findings:

1. Click the finding in the Findings table.
2. View the recommendation and details for the selected finding in the Recommendation Details and Finding Details section.



See Also:

"Performance Self-Diagnostics: Automatic Database Diagnostic Monitor "

Using Advisors to Optimize Database Performance

Oracle Database includes a set of *advisors* to help you manage and tune your database. This section contains background information about these advisors and instructions for their use. The following topics are covered:

- About Advisors
- About the SQL Tuning Advisor
- About the Automatic SQL Tuning Advisor
- Configuring the Automatic SQL Tuning Advisor
- Viewing Automatic SQL Tuning Results
- Running the SQL Tuning Advisor
- Optimizing Memory Usage with the Memory Advisors

See Also:

Oracle Database 2 Day + Performance Tuning Guide

About Advisors

Advisors are powerful tools for database management. They provide specific advice on how to address key management challenges, covering a wide range of areas

including space, performance, and undo management. In general, advisors produce more comprehensive recommendations than alerts. This is because alert generation is intended to be low cost and have minimal impact on performance, whereas advisors consume more resources and perform more detailed analysis. This, along with the what-if capability of some advisors, provides vital information for tuning that cannot be procured from any other source. Some advisors are run automatically.

Advisors are provided to help you improve database performance. These advisors include Automatic Database Diagnostic Monitor (ADDM), SQL advisors, and memory advisors. For example, the SGA Advisor graphically displays the impact on performance of changing the size of the System Global Area (SGA).

You can run a performance advisor when faced with the following situations:

- You want to resolve a problem in a specific area, for example, to determine why a given SQL statement is consuming 50 percent of CPU time and what to do to reduce its resource consumption.You can use the SQL Tuning Advisor.
- You are planning to add memory to your system. You can use the Memory Advisor to determine the database performance impact of increasing your SGA or PGA (Program Global Area).

You can also invoke some of the advisors from the Performance Hub page, or through recommendations from ADDM.

Table 2 describes the performance advisors.

Table 2 Performance Advisors

Advisor	Description
Automatic Database Diagnostics Monitor (ADDM)	ADDM makes it possible for Oracle Database to diagnose its own performance and determine how any identified problems can be resolved.See "Performance Self-Diagnostics: Automatic Database Diagnostic Monitor " and "Diagnosing Performance Problems Using ADDM."
SQL Tuning Advisor	The SQL Tuning Advisor analyzes one or more SQL statements and makes recommendations for improving performance. This advisor is run automatically during the maintenance periods, but can also be run manually. See "About the Automatic SQL Tuning Advisor" and "Running the SQL Tuning Advisor." For more information about the maintenance windows (time periods) for your database, see *Oracle Database Reference*.
Memory Advisors • Memory Advisor • SGA Advisor • Buffer Cache Advisor • PGA Advisor	The Memory Advisors provide graphical analyses of total memory target settings, SGA and PGA target settings, or SGA component size settings. You use these analyses to tune database performance and for what-if planning. Depending on the current memory management mode, different memory advisors are available. • If Automatic Memory Management is enabled, the Memory Advisor is available. This advisor provides advice for the total memory target for the instance. • If Automatic Shared Memory Management is enabled, then the SGA Advisor and PGA Advisor are available. • If Manual Shared Memory Management and Automatic PGA Memory are enabled, then the Buffer Cache Advisor and PGA Advisor are available.

Advisor	Description
	See "Optimizing Memory Usage with the Memory Advisors" for more information about memory advisors, and see "Managing Memory" for more information about memory management modes.
Undo Advisor	The Undo Advisor assists in correctly sizing the undo tablespace. The Undo Advisor can also be used to set the low threshold value of the undo retention period for any Oracle Flashback requirements. See "Computing the Minimum Undo Tablespace Size Using the Undo Advisor."

See Also:

Oracle Database 2 Day + Performance Tuning Guide

About the SQL Tuning Advisor

The SQL Tuning Advisor examines a given SQL statement or a set of SQL statements and provides recommendations to improve efficiency. It can make various types of recommendations, such as creating a **SQL profile** (a collection of information that enables the query optimizer to create an optimal execution plan for a SQL statement), restructuring SQL statements, and refreshing optimizer statistics. SQL Tuning Advisor also enables you to pick an alternative execution plan (stored in AWR) from the past and use it with the SQL statement, and can also recommend degree of parallelism profiles. EM Express enables you to accept and implement many of these recommendations with just a few mouse clicks.

You use the SQL Tuning Advisor to tune a single SQL statement or multiple SQL statements. Typically, you run the SQL Tuning Advisor in response to an ADDM performance finding that recommends its use. You can also run it periodically on the most resource-intensive SQL statements, and on a SQL workload.

When tuning multiple SQL statements, the SQL Tuning Advisor does not recognize interdependencies between the SQL statements. It solves SQL performance problems by identifying problems with individual SQL statements, such as a poorly performing optimizer plan or the mistaken use of certain SQL structures.

You can run the SQL Tuning Advisor against the following sources:

- Activity—The most resource-intensive SQL statements executed during the last hour that appear on the Activity tab of the Performance Hub that might have caused recent performance problems.
- Historical SQL—A SQL statement from the last day, week, or month that appears on the Activity tab of the Performance Hub when one of the historical settings is selected in the **Select Time Period** field. Use this option for proactive tuning of SQL statements.
- Historical SQL from ADDM—A resource-intensive SQL statement from an ADDM task that you discover when analyzing a task on the ADDM tab of the Performance Hub.
- SQL statement in SQL Tuning Advisor—A resource-intensive SQL statement that appears as a tuning task in SQL Tuning Advisor.

- SQL tuning sets (STS)—A set of SQL statements you provide. An STS can be created from SQL statements captured by AWR snapshots or from a SQL workload.

Note:

You cannot create an STS using EM Express. See *Oracle Database SQL Tuning Guide* for information on creating an STS.

See Also:

- *Oracle Database 2 Day + Performance Tuning Guide* for more information about tuning SQL statements with the SQL Tuning Advisor
- "Running the SQL Tuning Advisor"

About the Automatic SQL Tuning Advisor

The SQL Tuning Advisor runs automatically during system maintenance windows (time periods) as a maintenance task. During each automatic run, the advisor selects high-load SQL queries in the system and generates recommendations on how to tune these queries.

The Automatic SQL Tuning Advisor can be configured to automatically implement SQL profile recommendations. A **SQL profile** contains additional SQL statistics that are specific to the SQL statement and enable the query optimizer to generate a significantly better execution plan at run time. If you enable automatic implementation, then the advisor creates SQL profiles for only those SQL statements where the performance improvement would be at least threefold. Other types of recommendations, such as the creation of new indexes, refreshing optimizer statistics, or restructuring SQL, can only be implemented manually. DML statements are not considered for tuning by the Automatic SQL Tuning Advisor.

You can view a summary of the results of automatic SQL tuning, and a detailed report about recommendations made for all SQL statements that the SQL Tuning Advisor has processed. You can then implement selected recommendations. You can also view the recommendations that were automatically implemented.

You can disable the Automatic SQL Tuning Advisor, if desired.

See Also:

- "Viewing Automatic SQL Tuning Results"
- "Configuring the Automatic SQL Tuning Advisor"
- *Oracle Database 2 Day + Performance Tuning Guide*
- *Oracle Database Administrator's Guide* for more information about automated maintenance tasks

Configuring the Automatic SQL Tuning Advisor

The following are some configuration tasks that you might want to perform for the Automatic SQL Tuning Advisor:

- Enable automatic implementation of SQL profile recommendations.

  Automatic implementation is disabled by default.

- Change the maximum number of SQL profiles implemented during one run of the SQL Tuning Advisor

  When automatic implementation of SQL profile recommendations is enabled, 20 SQL profiles are implemented during a run of the SQL Tuning Advisor, by default.

- Change the maximum number of SQL profiles that can be implemented overall.

  When automatic implementation of SQL profile recommendations is enabled, a total of 10000 SQL profiles can be implemented by SQL Tuning Advisor, by default.

To configure the Automatic SQL Tuning Advisor:

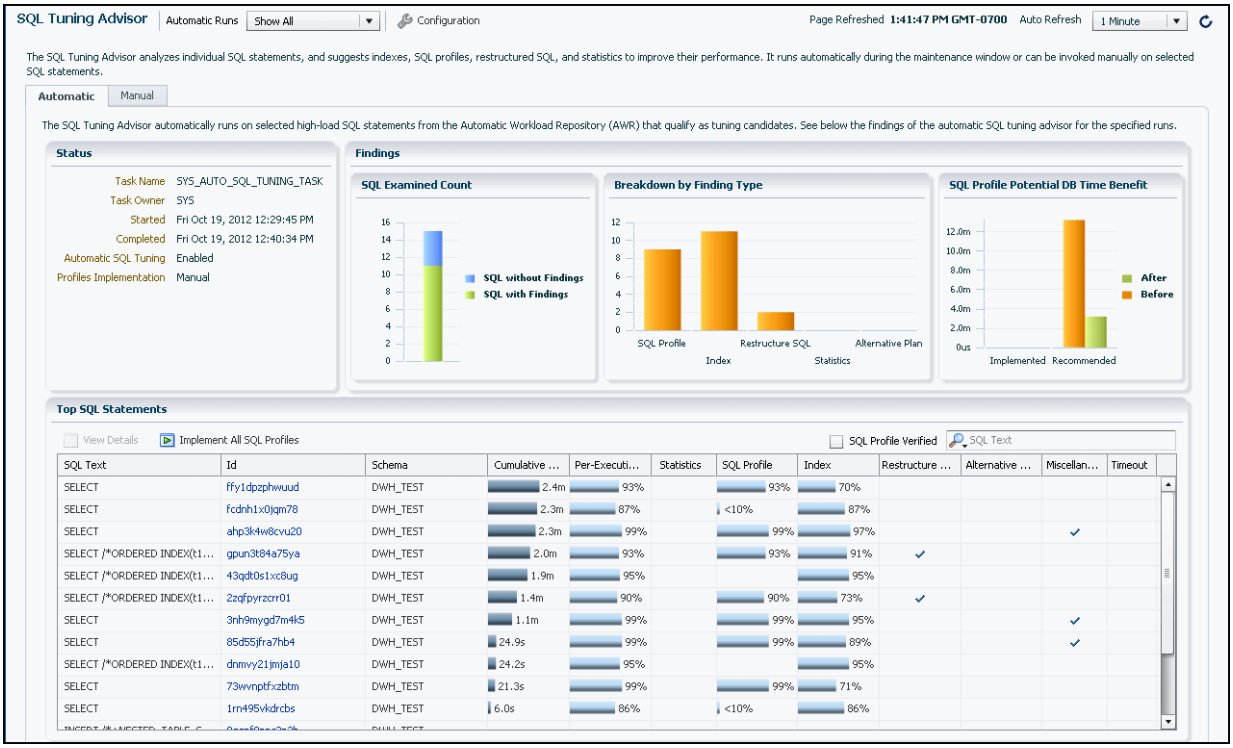
1. Log into EM Express as a user who has the `EM_EXPRESS_ALL` role.

   See "Granting Access to EM Express for Nonadministrative Users" for more information about the `EM_EXPRESS_ALL` role.

2. In EM Express, from the **Performance** menu, select **SQL Tuning Advisor**.

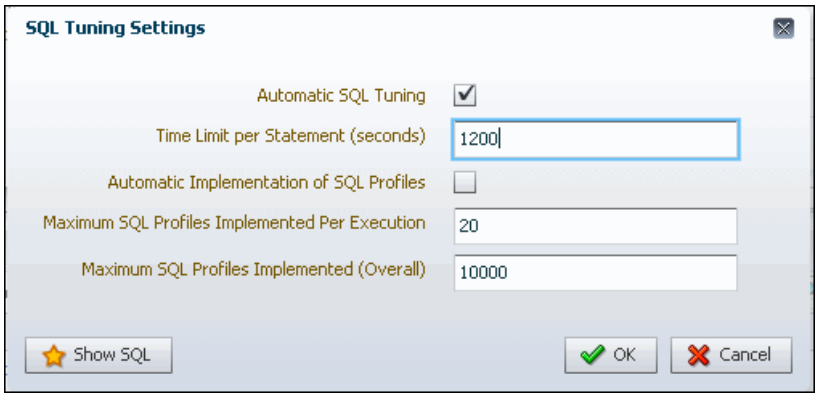
   The SQL Tuning Advisor page appears, with the Automatic tab showing.

   The SQL tuning task that appears on the Automatic tab is the `SYS_AUTO_SQL_TUNING_TASK`. This tuning task is created daily by the Automatic SQL Tuning Advisor. The task includes any high-load SQL queries for which the Automatic SQL Tuning Advisor has generated tuning recommendations.



3. Click the **Configuration** button.

   The SQL Tuning Settings dialog box appears.



4. (Optional) To disable the Automatic SQL Tuning Advisor, remove the check mark for the the **Automatic SQL Tuning** option.

   Note:

   If you disable the **Automatic SQL Tuning** option, none of the other options in the SQL Tuning Settings dialog box have any effect.

5. (Optional) In the **Time Limit per Statement (seconds)** field, enter the maximum time that SQL Tuning Advisor should take to tune any single SQL statement (in seconds).

6. (Optional) In the **Automatic Implementation of SQL Profiles** field, enter a check mark to enable the automatic implementation of SQL profiles, or remove the check mark to disable the automatic implementation of SQL profiles.

   Note:

   If you disable the **Automatic Implementation of SQL Profiles** option, the settings for the **Maximum SQL Profiles Implemented per Execution** and **Maximum SQL Profiles Implemented (Overall)** options have no effect.

7. (Optional) In the **Maximum SQL Profiles Implemented per Execution** field, enter the total number of SQL profiles that can be implemented during a single daily run of the SQL Tuning Advisor. The default value is 20.

8. (Optional) In the **Maximum SQL Profiles Implemented (Overall)** field, enter the total number of SQL profiles that can be implemented overall. The default value is 10000.

9. Click **OK**.

   A confirmation page appears.

   See Also:

   "About the Automatic SQL Tuning Advisor" for more information about SQL profiles

Viewing Automatic SQL Tuning Results

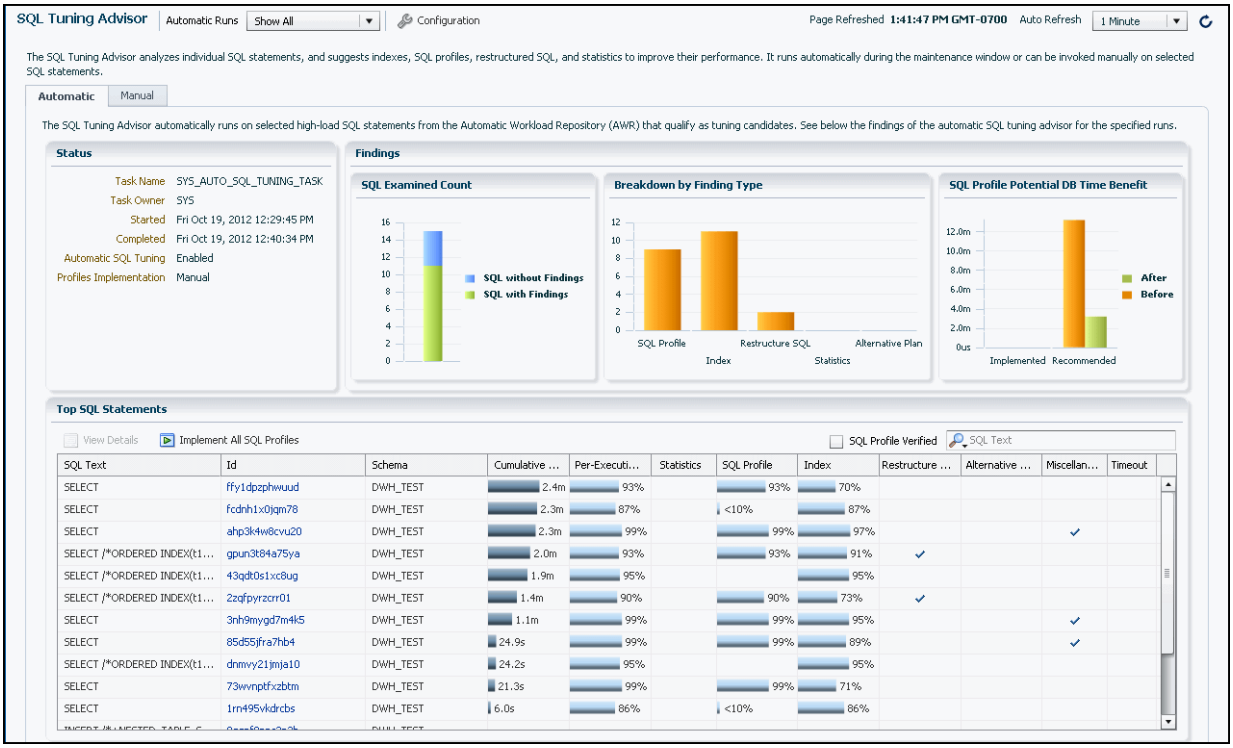
You can track the activities of the Automatic SQL Tuning Advisor with EM Express.

To view automatic SQL tuning results:

1. In EM Express, from the **Performance** menu, choose **SQL Tuning Advisor**.

   The SQL Tuning Advisor page appears, with the Automatic tab showing.

   The SQL tuning task that appears on the Automatic tab is the `SYS_AUTO_SQL_TUNING_TASK`. This tuning task is created daily by the Automatic SQL Tuning Advisor. The task includes any high-load SQL queries for which the Automatic SQL Tuning Advisor has generated tuning recommendations.

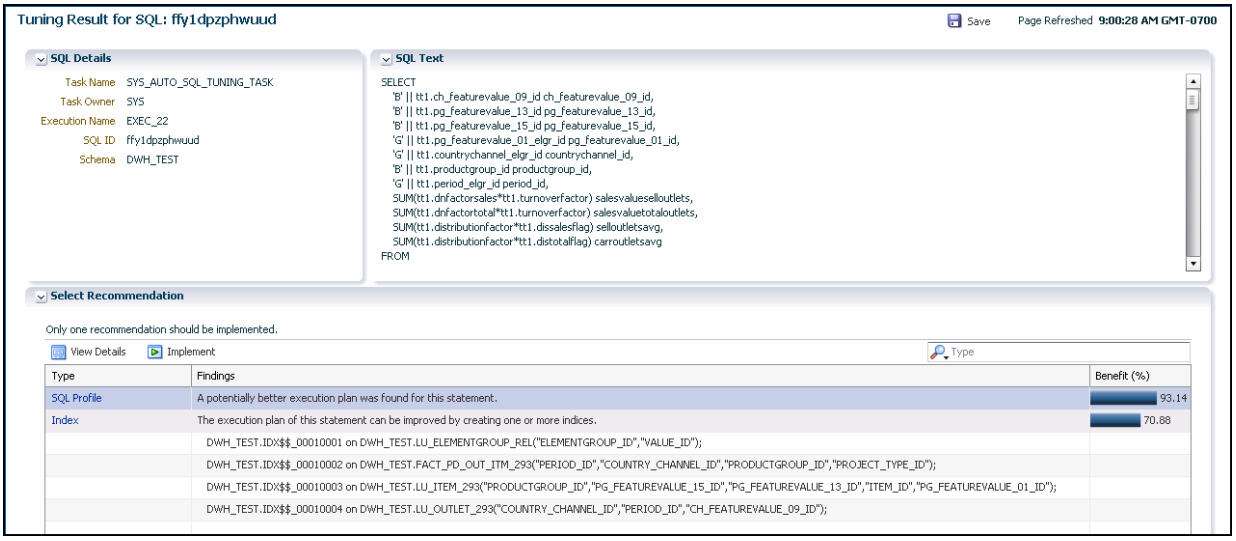


If you configured Automatic SQL Tuning Advisor to automatically implement SQL profile recommendations, then the SQL Profile Potential DB Time Benefit chart on the Automatic tab of the SQL Tuning Advisor page will include an Implemented bar. Click the Implemented bar to see all the SQL profiles that were automatically implemented.

2. In the Top SQL Statements table, select a row that includes a SQL statement for which you want to view tuning recommendations, and then click **View Details**.

   In this example, the SELECT statement with a SQL ID of `ffy1dpzphwuud` was selected and **View Details** was clicked.

   The Tuning Result for SQL page appears, which shows a summary of the tuning recommendations for the selected SQL statement.



3. The Select Recommendation section advises that only one recommendation on the page should be implemented.

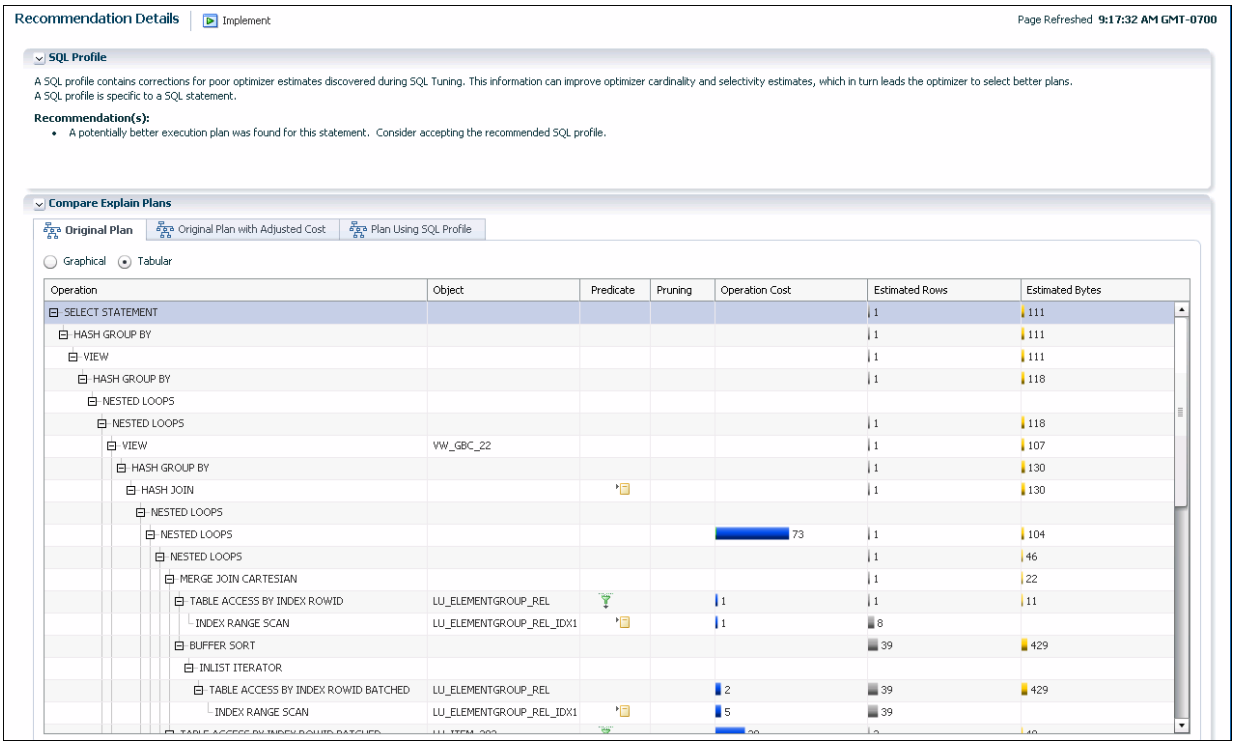
   To implement a recommendation, select it in the table and click **Implement**. You will be prompted to provide the necessary information to implement the recommendation.

   To help you decide which (if any) of the recommendations to implement, you may want to view more details about each of the recommendations.

   To view more details about a recommendation, select it in the table, then click **View Details**. In this example, the SQL Profile recommendation is selected.

   The Recommendation Details page appears.

4. The top section of this page describes the performance recommendation, and the section is named after the type of recommendation. Some possible names for this section are "Stale or Missing Statistics," "Restructure SQL," and "SQL Profile." This section provides an overview of the recommendation.



The Compare Explain Plans section at the bottom of the page includes tabs that that enable you to view one or more execution plans for the selected statement. The four tabs that can appear are the Original Plan, Original Plan with Adjusted Cost, Plan Using SQL Profile, and Alternative Plan tabs. The Graphical and Tabular buttons enable you to display an execution plan in graphical or tabular format. In this example, the execution plan is displayed in tabular format.

For recommendations that do not include a potentially better execution plan, only the Original Plan tab appears, and the operations for the original plan are shown on the tab.

When you click the Original Plan with Adjusted Cost tab, the execution plan steps are the same as the Original Plan steps, but the Original Plan with Adjusted Cost steps have different costs for the steps (as shown in the **Operation Cost** column).

If you click the Plan Using SQL Profile tab, the steps are different than the Original Plan steps, and the steps have different costs (as shown in the **Operation Cost** column).

The Alternative Plan button appears when the execution history for the original plan cannot be found. In this case, if you know that the alternative plan suggested by SQL Tuning Advisor is better than the original plan, you can create a SQL plan baseline for the alternative plan so that the Oracle optimizer will pick the alternative plan for the statement in the future.

Click the **Implement** button at the top of the Recommendation Details page to implement a recommendation.

See Also:

- *Oracle Database 2 Day + Performance Tuning Guide*
- "About the Automatic SQL Tuning Advisor"

Running the SQL Tuning Advisor

Use the SQL Tuning Advisor for tuning SQL statements. Typically, you run this advisor in response to an ADDM performance finding that recommends its use. You can also start the SQL Tuning Advisor manually. One reason is to tune statements that the Automatic SQL Tuning Advisor has not considered for tuning.

As described in "About the SQL Tuning Advisor," the SQL Tuning Advisor can select SQL statements to tune from several sources. The following scenario assumes that you want to tune the SQL statements with the most activity:

To run the SQL Tuning Advisor:

1. In EM Express, from the **Performance** menu, choose **Performance Hub**.

   The Performance Hub page appears.

2. In the **Select Time Period** field, select the desired time period.

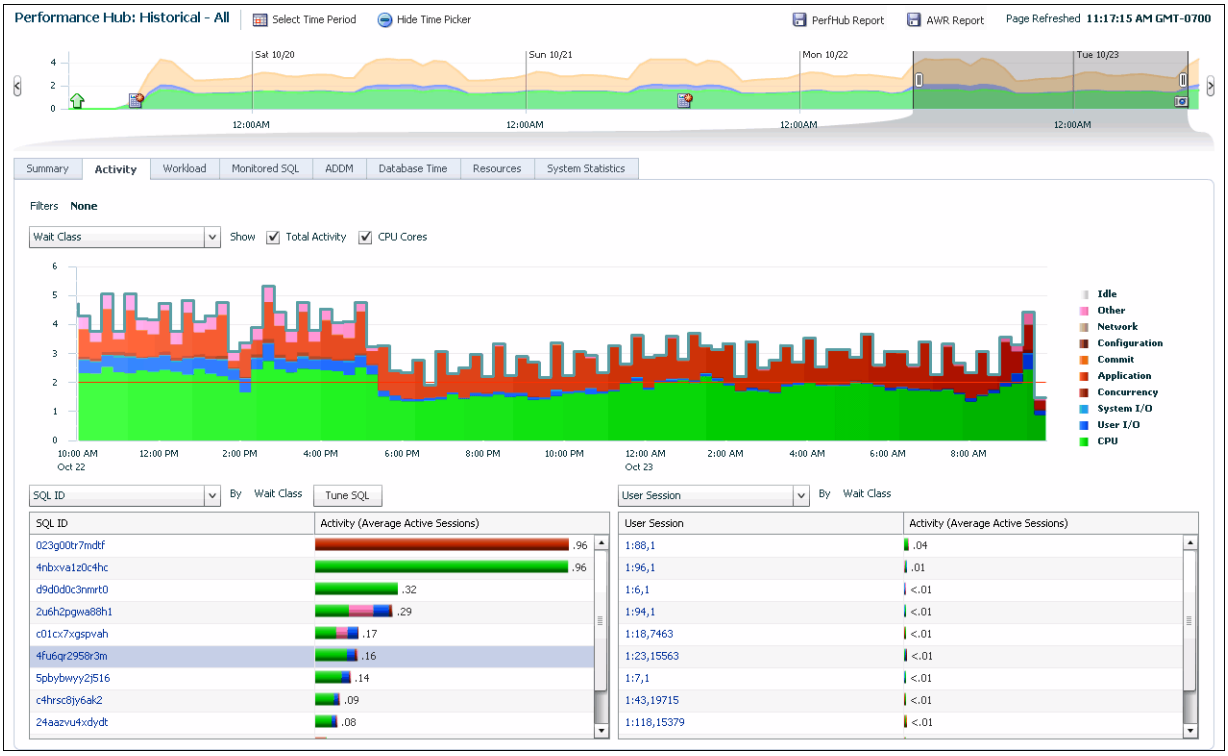
   In this example, **Historical - All** has been selected in the **Select Time Period** field.

3. Select **Activity**.

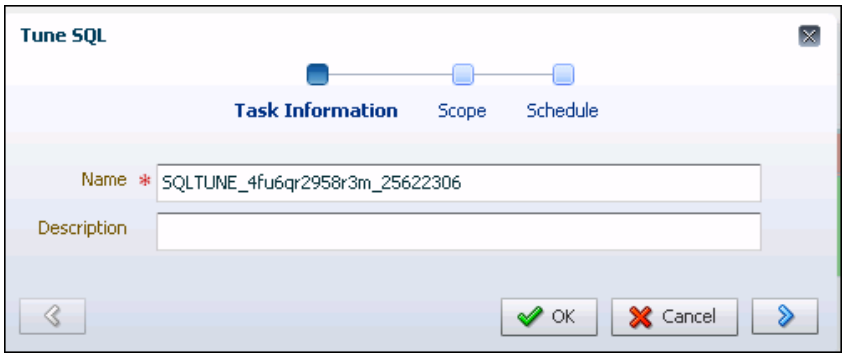
   The Activity tab appears.

4. In the table at the bottom of the Activity tab, select the row that includes the SQL statement that you want to tune, and then click the **Tune SQL** button.

   In this example, the SQL statement in the sixth row of the table is selected.



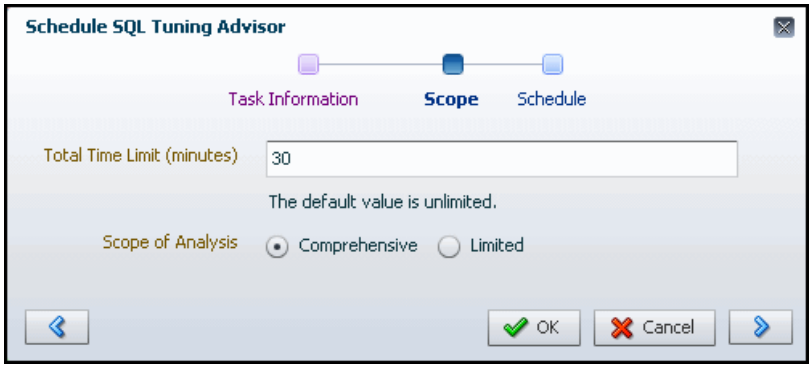
5. The Schedule SQL Tuning Advisor wizard appears, with the Task Information page showing.



On the Task Information page, you can accept the tuning task name generated by the system, or enter a name of your choosing for the tuning task that will be created for the selected SQL statement. You also have the option of entering a description for the tuning task.

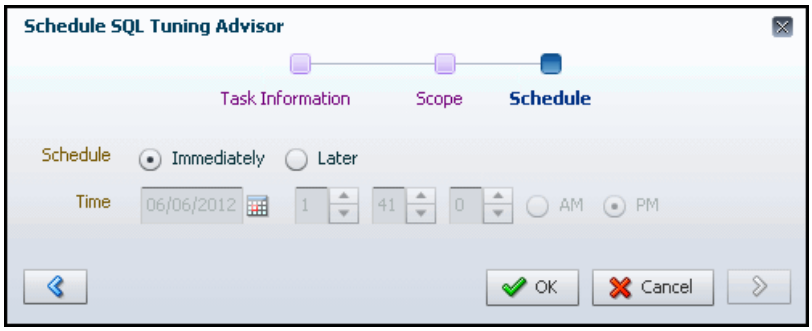
Click the right arrow button.

6. The Scope page appears. Specify the total time SQL Tuning Advisor should spend analyzing the statement (the default value is Unlimited), and the scope of the analysis (Comprehensive or Limited).



Click the right arrow button.

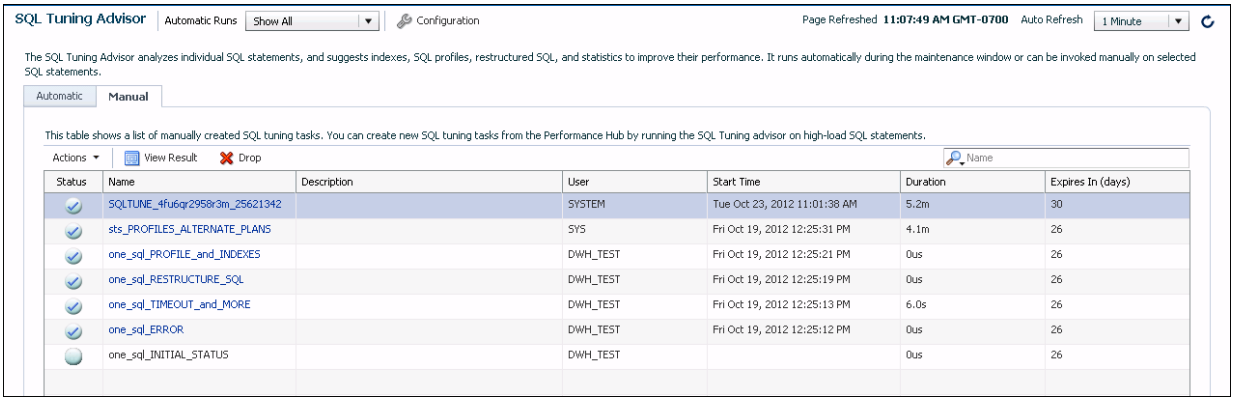
7. The Schedule page appears. On this page, you can schedule SQL Tuning Advisor to analyze the statement immediately or later. If you choose to have SQL Tuning Advisor analyze the statement later, specify the time that you want SQL Tuning Advisor to analyze the statement.



Click **OK** to begin the tuning task creation by SQL Tuning Advisor for the selected SQL statement.

8. When the SQL Tuning Advisor finishes analyzing the SQL statement, the **Completed** (check mark) icon appears in the **Status** column on the SQL Tuning Advisor page.

   Select the row that includes the SQL statement that SQL Tuning Advisor has finished analyzing, and then click **View Result** to see the recommendations the SQL Tuning Advisor has for this SQL statement. In this example, the first row is selected.



The Tuning Result for SQL: SQLID page appears, which shows a summary of the tuning recommendations for the selected SQL statement.

9. The Select Recommendation section at the bottom of the page shows the recommendations for tuning the SQL statement.

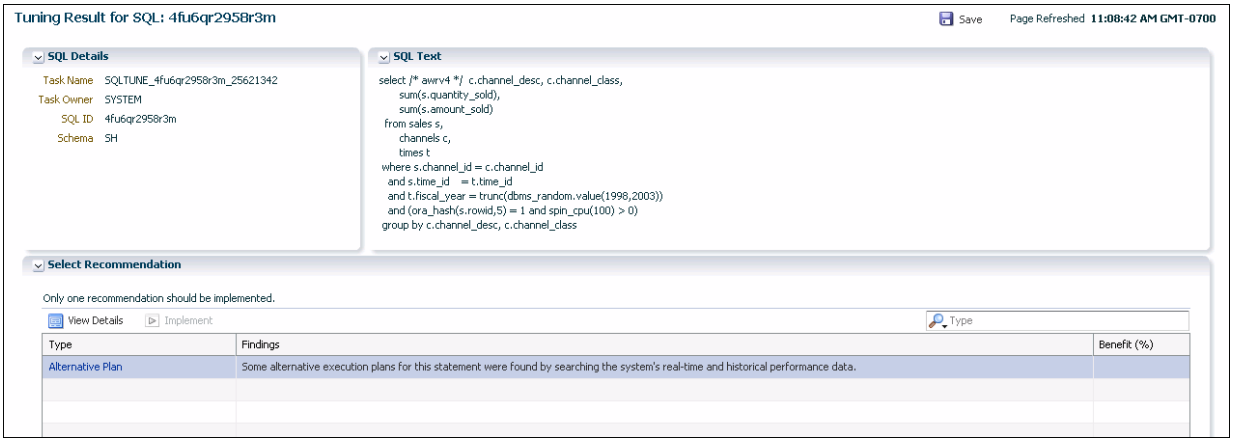
   If there are multiple recommendations on the page, only one of them should be implemented.

   To implement a recommendation, select it in the table and click **Implement**. You will be prompted to provide the necessary information to implement the recommendation.

   To help you decide which (if any) of the recommendations to implement, you may want to view more details about each of the recommendations.

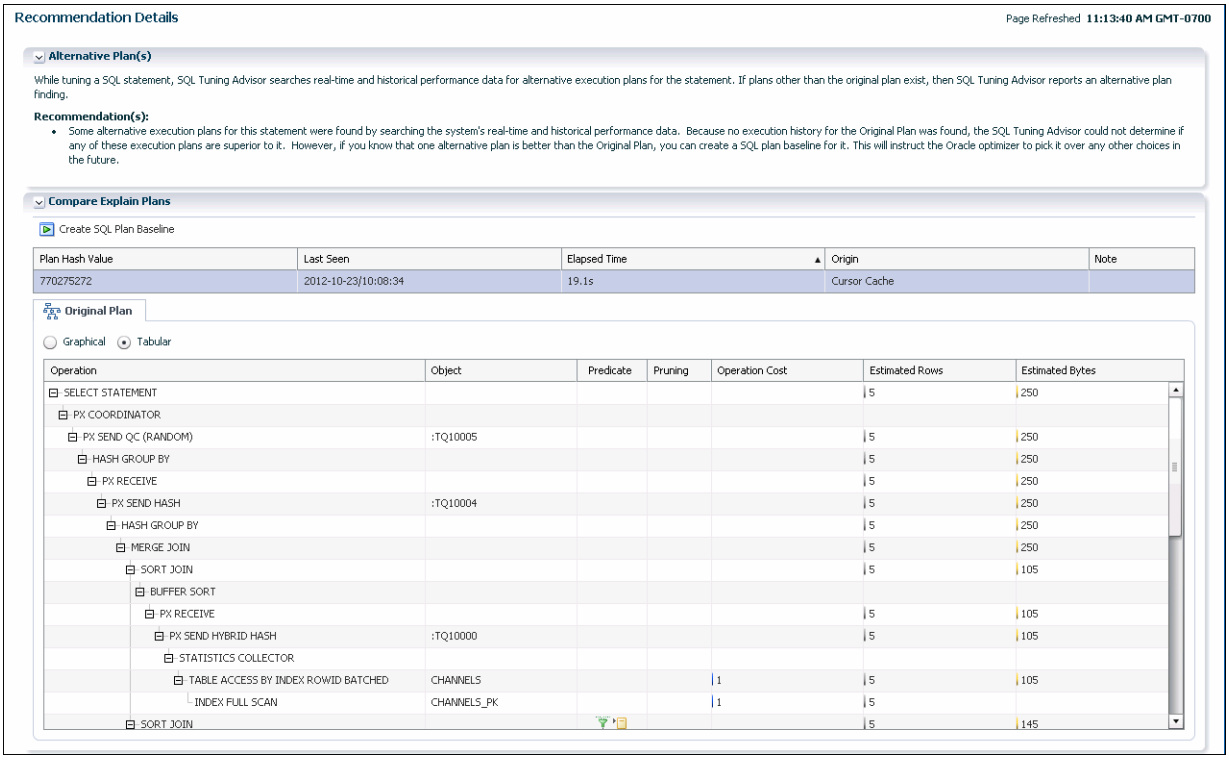
   To view more details about a recommendation, select it in the table, then click **View Details**.

   In this example, the **Some alternative execution plans for this statement were found by searching the system's real-time and historical performance data** recommendation is selected.



The Recommendation Details page appears.

10. The top section of this page describes the performance recommendation, and the section is named after the type of recommendation. Some possible names for this section are "Alternative Plan(s)," "Stale or Missing Statistics," "Restructure SQL," and "SQL Profile." This section provides an overview of the recommendation.



The Compare Explain Plans section at the bottom of the page includes one or more tabs that that enable you to view one or more execution plans for the selected statement. The four tabs that can appear are the Original Plan, Original Plan with Adjusted Cost, Plan Using SQL Profile, and Alternative Plan tabs. The Tabular and Graphical button enable you to display an execution plan in tabular or graphical format. In this example, the execution plan is displayed in tabular format.

For recommendations that do not include a potentially better execution plan, only the Original Plan tab appears, and the operations for the original plan are shown on the Original Plan subpage.

On the Original Plan with Adjusted Cost subpage, the execution plan steps are the same as the Original Plan steps, but the Original Plan with Adjusted Cost steps have different costs for the steps (as shown in the **Operation Cost** column).

On the Plan Using SQL Profile subpage, the steps are different than the Original Plan steps, and the steps have different costs (as shown in the **Operation Cost** column).

The Alternative Plan subpage is available when the execution history for the original plan cannot be found. In this case, if you know that the alternative plan suggested by SQL Tuning Advisor is better than the original plan, you can click the **Create SQL Plan Baseline** button to create a SQL plan baseline for the alternative plan so that the Oracle optimizer will pick the alternative plan for the statement in the future.

To implement a recommendation, click the **Back** button in your browser, and implement the recommendation on the Tuning Result page.

See Also:

- "About the SQL Tuning Advisor"
- "About the Automatic SQL Tuning Advisor"

Optimizing Memory Usage with the Memory Advisors

This section contains:

- About the Memory Advisors

See Also:

- "Managing Memory"
- "About Advisors"

About the Memory Advisors

Adequate physical memory has a significant impact on the performance of your Oracle Database. With its automatic memory management capabilities, Oracle Database can automatically adjust the memory distribution among the various SGA and PGA components for optimal performance. These adjustments are made within the boundaries of the total amount of memory that you allocate to the database.

ADDM periodically evaluates the performance of your database to determine performance problems. If ADDM finds that the current amount of available memory is inadequate and adversely affecting performance, then it can recommend that you increase memory allocations. You can select new memory allocations using the Memory Advisors.

Additionally, you can use the Memory Advisors to perform what-if analysis on the following:

- The database performance benefits of adding physical memory to your database
- The database performance impact of reducing the physical memory available to your database

With the Memory Advisors, you can obtain memory sizing advice as follows:

- If automatic memory management is enabled, you can get a prediction of the percentage of time saved by using a different target memory size setting for the Oracle instance.

  See "Modifying Memory Settings – Automatic Memory Management" for more information.

- If automatic memory management is disabled and automatic shared memory management is enabled, you can get a prediction of the percentage of time saved by using a different total SGA size.

  See "Modifying Memory Settings – Automatic Shared Memory Management" for more information.

- If only manual shared memory management is enabled, then you can get a prediction of the percentage of reads saved by using a different database cache size.

  See "Modifying Memory Settings - Manual Shared Memory Management" for more information.

Monitoring and Tuning: Oracle by Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this section, and includes annotated screenshots.

To view the Monitoring and Tuning the Database OBE,enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6290,1
```

11

Managing PDBs with EM Express

This chapter describes managing pluggable databases (PDBs) in a multitenant container database (CDB) using Oracle Enterprise Manager Database Express (EM Express).

This chapter covers the following topics:

- Getting Started
- Overview of CDBs and PDBs
- Accessing the Containers Page
- Modifying Resource Plans for a PDB
- Setting Storage Limits for a PDB
- Configuring Oracle Managed Files
- Provisioning a PDB
- Removing PDBs
- Opening PDBs
- Closing PDBs

A different HTTPS port must be configured for each CDB and PDB that you want to manage using EM Express.

See Also:

- "Starting EM Express for a CDB" for information about starting EM Express to manage a CDB
- "Starting EM Express for a PDB" for information about starting EM Express to manage a PDB

Getting Started With Managing PDBs Using EM Express

This section helps you get started with this chapter by providing an overview of the features EM Express provides for creating and managing the PDBs in a CDB.

Click the links in Table 1 to go to the sections that provide more information.

Table 1 Getting Started with Managing PDBs in a CDB with EM Express

Feature	More Information
Overview of CDBs and PDBs	See "Overview of CDBs and PDBs" for a conceptual overview of CDBs and PDBs. For detailed conceptual information about CDBs and PDBs, see *Oracle Database Concepts* and *Oracle Database Administrator's Guide*.
The Containers page in EM Express	See "Accessing the Containers Page."
Modifying resource plans for a CDB	See "Using EM Express to Modify a CDB Resource Plan."
Setting resource limits for a PDB	See "Using EM Express to Set Resource Limits for a PDB."
Setting storage limits for a PDB	See "Setting Storage Limits for a PDB."
Configuring Oracle Managed Files for a CDB	See "Configuring Oracle Managed Files." For detailed information about Oracle Managed Files, see *Oracle Database Administrator's Guide*.
Creating a new PDB from the seed	See "Creating a New PDB from the Seed."
Creating a PDB by cloning a PDB in the same CDB	See "Creating a PDB by Cloning a PDB in the Same CDB."
Creating a PDB by cloning a PDB from a remote CDB	See "Creating a PDB by Cloning a PDB from a Remote CDB."
Plugging in an unplugged PDB	See "Plugging in an Unplugged PDB."
Unplugging a PDB	See "Unplugging a PDB."
Dropping a PDB	See "Dropping a PDB."
Opening a PDB	See "Opening a PDB."
Opening all the PDBs in a CDB	See "Opening All the PDBs in a CDB."
Closing a PDB	See "Closing a PDB."
Closing all the PDBs in a CDB	See "Closing All the PDBs in a CDB."

Overview of CDBs and PDBs

An Oracle Database can contain a portable collection of schemas, schema objects, and nonschema objects that appear to an Oracle Net client as a separate database. This self-contained collection is called a PDB. A CDB can include zero, one, or more PDBs. Oracle Database 12*c* allows you to create many PDBs within a single CDB. Applications that connect to databases view PDBs and earlier versions of Oracle Database in the same manner.

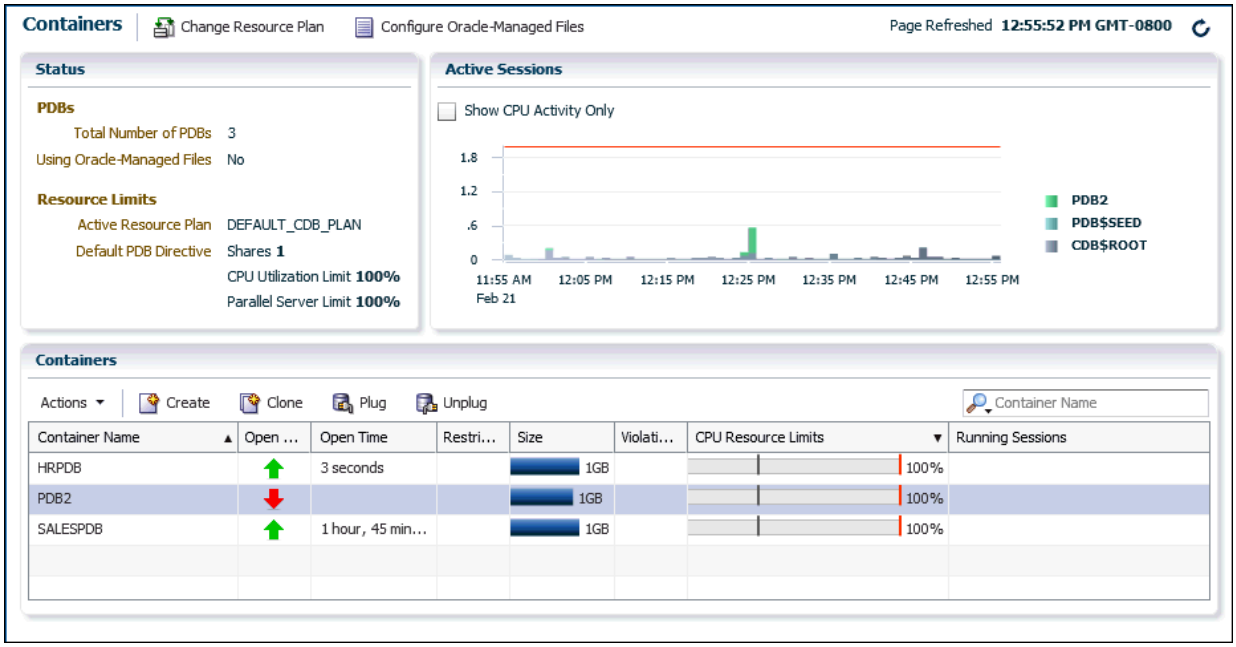
EM Express enables database administrators to manage a CDB and its PDBs.

See Also:

Oracle Database Concepts and *Oracle Database Administrator's Guide* for detailed conceptual information about CDBs and PDBs

Accessing the Containers Page in EM Express

Many of the features that EM Express provides for managing PDBs in a CDB are available on the Containers page of EM Express:



EM Express provides the Containers page only for CDBs. The Containers page is not provided for non-CDBs.

To access the Containers page for a CDB:

1. In EM Express, navigate to the Database Home page for the CDB.

   For more information, see "Accessing the Database Home Page."

2. In the Status section of the Database Home page, click the **CDB (n PDBs)** link (where **n** is the number of PDBs in the CDB) to go to the Containers page for the CDB.

The features that EM Express provides for managing PDBs on the Containers page are described in more detail later in this chapter.

Modifying Resource Plans for a PDB Using EM Express

This section describes how to use EM Express to make changes to a resource plan for a CDB and to limit the resources to allocate to a PDB.

This section assumes that you understand how to use Resource Manager with CDBs and PDBs.

It includes the following sections:

Using Resource Manager to Create CDB and PDB Resource Plans

You can use Oracle Database Resource Manager (Resource Manager) to allocate resources to PDBs in a CDB.

In a CDB, you can have multiple workloads within multiple PDBs competing for system and CDB resources.

In a CDB, Resource Manager can manage resources on two basic levels:

- **CDB level** - Resource Manager can manage the workloads for multiple PDBs that are contending for system and CDB resources. You can specify how resources are allocated to PDBs, and you can limit the resource utilization of specific PDBs.
- **PDB level** - Resource Manager can manage the workloads within each PDB.

After you have used Resource Manager to create a resource plan for a CDB, you can use EM Express to make changes to the CDB resource plan and to limit the resources to allocate to a PDB.

Note:

See *Oracle Database Administrator's Guide* for information on using Resource Manager with CDBs and PDBs

---

Using EM Express to Modify a CDB Resource Plan

This section provides information about modifying a resource plan for a CDB.

Note:

Make sure that a resource plan exists for the PDB before you attempt to modify the plan.

To modify the resource plan for a CDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the resource plan that you want to modify.

   For more information, see "Accessing the Containers Page."

2. Near the top of the Containers page, click the **Change Resource Plan** button.
3. In the Change Resource Plan wizard, supply values for these fields:

   - **Current Active Resource Plan**: Choose the CDB resource plan you want to modify.

   In the Default PDB Directive section, make changes to the default directives for the PDBs:

- **Shares**: Assign a new default share value for the PDBs, or keep the current value.
- **CPU Utilization Limit (%)**: Assign a new default CPU utilization percentage value for the PDBs, or keep the current value.
- **Parallel Server Limit (%)**: Assign a new default parallel server utilization percentage value for the PDBs, or keep the current value.

4. Click **OK**.

   In the Status section of the Containers page, the Resource Limits values are updated to show the changes you made in the Change Resource wizard.

Using EM Express to Set Resource Limits for a PDB

This section provides information about setting resource limits for a PDB.

Note:

Before you set resource limits for a PDB:

- A resource plan must be created for the CDB.
- The CDB must contain at least one PDB whose resource usage you want to limit.

To set resource limits for a PDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the PDB whose resource usage you want to limit.

   For more information, see "Accessing the Containers Page."

2. In the Containers section, click the name of the PDB whose resource usage you want to limit, and then choose **Set Resource Limits** from the **Actions** menu.

3. In the Set Resource Limits wizard, supply values for these fields:

   - **Shares**: Assign a new share value for this PDB, or keep the current value.
   - **CPU Utilization Limit (%)**: Assign a new CPU utilization percentage value for this PDB, or keep the current value.
   - **Parallel Server Limit (%)**: Assign a new parallel server utilization percentage value for this PDB, or keep the current value.

4. Click **OK**.

   The Confirmation box advises you that the resource limits for the PDB were set successfully. In the Containers section of the Containers page, move your cursor over the CPU Resource Limits column for the PDB to see the updated values that you specified for the PDB in the Set Resource Limits wizard.

Setting Storage Limits for a PDB Using EM Express

This section provides information on setting storage limits for a PDB.

Note:

Before you set storage limits for a PDB:

- The CDB must contain at least one PDB.
- The PDB must be open in read write mode.

To set storage limits for a PDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the PDB whose storage limits you want to set.

   For more information, see "Accessing the Containers Page."

2. In the Containers section, click the name of the PDB whose storage limits you want to set, and then choose **Set Storage Limits** from the **Actions** menu.

3. In the Set Storage Limits wizard, supply values for these fields:

   - **Max Size Unlimited**: Enable this option to enable unlimited storage for all the tablespaces for this PDB, or disable this value and then specify the maximum size for all the tablespaces in the **Max Size** field.
   - **Max Shared Temp Size Unlimited**: Enable this option to enable unlimited storage for the temporary tablespace shared by the sessions connected to the PDB, or disable this value and then specify the maximum size for the temporary tablespace for this PDB in the **Max Shared Temp Size** field.
   - Click **OK**.

     The Confirmation box advises you that the storage limits for the PDB were set successfully.

See Also:

"Opening a PDB"

Configuring Oracle Managed Files for a CDB Using EM Express

This section provides information on configuring Oracle Managed Files for a CDB.

Using Oracle Managed Files simplifies the administration of an Oracle Database. Oracle Managed Files eliminate the need for the DBA to directly manage the operating system files that comprise an Oracle Database. With Oracle Managed Files, you specify file system directories in which the database automatically creates, names, and manages files at the database object level.

Through initialization parameters, you specify the file system directory to be used for a particular type of file. The database then ensures that a unique file, an Oracle managed file, is created and deleted when no longer needed.

This feature does not affect the creation or naming of administrative files such as trace files, audit files, alert logs, and core files.

You can use EM Express to configure Oracle Managed Files for a CDB.

Note:

Before you configure Oracle Managed Files for a CDB:

- A CDB must exist.
- EM Express must be configured to manage the CDB.

To configure Oracle Managed Files for a CDB:

1. In EM Express, navigate to the Containers page for the CDB for which you want to configure Oracle Managed Files.

   For more information, see "Accessing the Containers Page."

2. Near the top of the Containers page, click the **Configure Oracle-Managed Files** button.

3. The Configure Oracle-Managed Files wizard enables you to set a value (a directory location) for the `DB_CREATE_FILE_DEST` initialization parameter. The directory you specify will be the destination of Oracle Managed Files.

   Specify values for these fields in the Configure Oracle-Managed Files wizard:

   - **Scope**: Specify **Memory** to make the change in memory, take effect immediately, and persist until the database is shut down. Choose **SPFile** to make the change in the server parameter file, so that the change will take place after the database is restarted. Choose both **Memory** and **SPFile** to change the value now and to have it remain in effect after the database is restarted.
   - **Deferred**: If specified, the deferred option allows to modify the value of the parameter only for future sessions that connect to the database. With deferred, current sessions retain the old value. If deferred is not specified, the value is changed immediately.
   - **Value**: Specify the directory to use for the `DB_CREATE_FILE_DEST` initialization parameter. This directory will be the destination of Oracle Managed Files.
   - **Comment**: Optionally, enter a comment regarding the changes you made.

4. Click **OK**.

   The Confirmation box advises you that the default directory for Oracle Managed Files has been set to the specified location. In the Status section of the Containers page, the **Using Oracle-Managed Files** field shows a value of `Yes`.

See Also:

- *Oracle Database Administrator's Guide* for more information on setting the `DB_CREATE_FILE_DEST` initialization parameter and on Oracle Managed Files
- *Oracle Database Reference* for more information about the `DB_CREATE_FILE_DEST` initialization parameter

Provisioning a PDB Using EM Express

You can provision PDBs by creating a new PDB within a CDB, by cloning an existing PDB, and by plugging an unplugged PDB into a CDB.

This section provides information about provisioning a PDB using EM Express. It includes the following topics:

Creating a New PDB from the Seed Using EM Express

This section provides information about creating a new PDB from the seed (PDB $SEED) using EM Express.

Note:

Before you create a new PDB from the seed:

- The CDB within which you want to create a PDB must exist, and EM Express must be configured to access the CDB.
- The CDB within which you want to create a PDB must be in read/write mode.
- The target host user must be the owner of the Oracle home that the CDB (within which you want to create the PDB) belongs to.

To create a new PDB from the seed:

1. In EM Express, navigate to the Containers page for the CDB in which you want to create the PDB.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, choose **Create** from the **Actions** menu. The Create PDB From Seed wizard appears.

3. On the General page of the Create PDB From Seed wizard, supply values for these fields:

   - **PDB Name**: Enter the name you want to use for the PDB you are creating.
   - **Username**: Enter the name of the administrative user who will manage the PDB you are creating.

     The username and password that you specify on this page are used to create the administrator as a local user in the PDB and grants the `PDB_DBA` role locally to the administrator. By default, no privileges are granted to the `PDB_DBA` role. You can grant roles or privileges to the `PDB_DBA` role once the PDB is created.

See Also:

Oracle Database Security Guide for more information about the PDB_DBA role and other predefined roles in an Oracle Database installation

- **Password**: Enter the password for the administrative user.
- **Confirm Password**: Enter the password for the administrative user again.

Click the right arrow button to go to the Storage page.

4. On the Storage page, select the type of location where you want to store the datafiles for the PDB:
   - If the target CDB (in which you are creating the PDB) is enabled with Oracle Managed Files and you want to use the same, then select **Use Oracle Managed Files (OMF)**.
   - If the target CDB does not use OMF, then specify a datafile location or accept the default specified in the **Datafile Location** field.
5. Also on the Storage page, choose whether or not to enable unlimited storage for the datafiles.

   If you do not enable unlimited storage, then you must specify values for these fields:

   - **Max Size**: The amount of storage that can be used by all tablespaces that belong to the PDB.
   - **Max Shared Temp Size**: The amount of storage in the default temporary tablespace shared by all PDBs that can be used by sessions connected to the PDB.

   If no values are specified for these fields, then their values will be set to unlimited when the PDB is created.
6. If the CDB has a current active resource plan, the Resource Limits page appears.

   Supply values for these fields, or keep the default values for the PDB you are creating:

   - **Shares**: Assign a new share value for this PDB, or keep the current value.
   - **CPU Utilization Limit (%)**: Assign a new CPU utilization percentage value for this PDB, or keep the current value.
   - **Parallel Server Limit (%)**: Assign a new parallel server utilization percentage value for this PDB, or keep the current value.
7. Click **OK**.

   The PDB is created and opened in read/write mode. EM Express adds the PDB to the list of containers that appears in the Containers section on the Containers page.

After configuring an HTTPS port for EM Express for this PDB, the PDB can be managed using EM Express. Administrative users who have been granted the EM_EXPRESS_ALL role in a PDB can use EM Express to manage the PDB.

See Also:

"Configuring the HTTPS Port for EM Express" for information on configuring an HTTPS port for EM Express for a PDB

Creating a PDB by Cloning a PDB in the Same CDB Using EM Express

This section provides information about creating a new PDB by cloning an existing PDB in the same CDB using EM Express.

Note:

Before you create a new PDB by cloning an existing PDB in the same CDB:

- The source PDB (the PDB that you want to clone) must exist, and EM Express must be configured to access the PDB.
- The source PDB must be open.
- The CDB (the CDB into which you want to plug in the cloned PDB) must exist, and EM Express must be configured to access the CDB.
- The CDB must be in read/write mode.
- The target host user must be the owner of the Oracle home that the CDB belongs to.

To create a new PDB by cloning an existing PDB in the same CDB:

1. In EM Express, navigate to the Containers page for the CDB in which you want to create the PDB.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, click the PDB that you want to clone, and then choose **Clone** from the **Actions** menu. The Clone wizard appears.

3. On the General page of the Clone wizard, enter the name you want to use for the PDB that will be created by the clone operation.

   Click the right arrow button to go to the Storage page.

4. On the Storage page, select the location where you want to store the datafiles for the PDB, or accept the default value.

5. Click **OK**.

   The PDB is created and opened in read/write mode. EM Express adds the PDB to the list of containers that appears in the Containers section on the Containers page.

After configuring an HTTPS port for EM Express for this PDB, the PDB can be managed using EM Express. Administrative users who have been granted the `EM_EXPRESS_ALL` role in a PDB can use EM Express to manage the PDB.

See Also:

- "Opening a PDB"
- "Configuring the HTTPS Port for EM Express" for information on configuring an HTTPS port for EM Express for a PDB

Creating a PDB by Cloning a PDB from a Remote CDB Using EM Express

This section provides information about creating a new PDB by cloning an existing PDB from a remote CDB using EM Express.

Note:

Before you create a new PDB by cloning an existing PDB from a remote CDB:

- The source PDB (the PDB that you want to clone) must exist, and EM Express must be configured to access the PDB.
- The source PDB must be open.
- The target CDB (the CDB into which you want to plug in the cloned PDB) must exist, and EM Express must be configured to access the CDB.
- The target CDB must be in read/write mode.
- The target host user must be the owner of the Oracle home that the CDB belongs to.

To create a new PDB by cloning an existing PDB from a remote CDB:

1. In EM Express, navigate to the Containers page for the CDB in which you want to create the PDB.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, **Remote Clone** from the **Actions** menu. The Clone a Remote PDB wizard appears.

3. On the General page of the Clone wizard, supply values for these fields:

   - **PDB Name**: Enter the name you want to use for the PDB that will be created by the clone operation.
   - **Source PDB Name**: Enter the name of the PDB that you want to clone in the remote CDB.
   - **DB Link**: Enter the name of the database link for the remote CDB that contains the PDB to be cloned.

   Click the right arrow button to go to the Storage page.

4. On the Storage page, select the type of location where you want to store the datafiles for the PDB:

- If the target CDB (in which you are creating the PDB) is enabled with Oracle Managed Files and you want to use the same, then select **Use Oracle Managed Files (OMF)**.
- If the target CDB does not use OMF, then specify a datafile location or accept the default specified in the **Datafile Location** field.

5. Also on the Storage page, choose whether or not to enable unlimited storage for the datafiles.

   If you do not enable unlimited storage, then you must specify values for these fields:

   - **Max Size**: The amount of storage that can be used by all tablespaces that belong to the PDB.
   - **Max Shared Temp Size**: The amount of storage in the default temporary tablespace shared by all PDBs that can be used by sessions connected to the PDB.

   If no values are specified for these fields, then their values will be set to unlimited when the PDB is created.

6. If the target CDB to which you are cloning the remote PDB has a current active resource plan, the Resource Limits page appears.

   Supply values for these fields, or keep the default values for the PDB you are creating:

   - **Shares**: Assign a new share value for this PDB, or keep the current value.
   - **CPU Utilization Limit (%)**: Assign a new CPU utilization percentage value for this PDB, or keep the current value.
   - **Parallel Server Limit (%)**: Assign a new parallel server utilization percentage value for this PDB, or keep the current value.

7. Click **OK**.

   The PDB is created and opened in read/write mode. EM Express adds the PDB to the list of containers that appears in the Containers section on the Containers page.

See Also:

"Opening a PDB"

Plugging in an Unplugged PDB Using EM Express

This section provides information about creating a new PDB by plugging in an unplugged PDB using EM Express.

Note:

Before you plug an unplugged PDB into a CDB:

- The target CDB (the CDB that you want to plug the unplugged PDB into) must exist, and EM Express must be configured to access the CDB.
- The target CDB must be in read/write mode.
- The XML file that describes the unplugged PDB and the other files associated with the unplugged PDB, such as the data files, must exist and must be readable.
- The target host user must be the owner of the Oracle home that the CDB (into which you want to plug the unplugged PDB) belongs to.
- The platforms of the source CDB host (the host on which the CDB that previously contained the unplugged PDB is installed) and the target CDB host (the host on which the target CDB is installed) must have the same endianness, and must have the same set of database options installed.
- The CDB that contained the unplugged PDB and the target CDB must have compatible character sets and national character sets. To be compatible, the character sets and national character sets must meet all the requirements specified in *Oracle Database Globalization Support Guide*.

To plug an unplugged PDB into a CDB:

1. In EM Express, navigate to the Containers page for the CDB that you want to plug the unplugged PDB into.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, choose **Plug** from the **Actions** menu. The Plug wizard appears.

3. On the General page of the Plug wizard, supply values for these fields:

   - **Metadata File**: Enter the full path to the metadata XML file that was created when the PDB was unplugged.
   - **Reuse PDB name from Metadata File**: Select this option, or disable it and specify an new name to use for the unplugged PDB when it is plugged into the CDB.
   - **Reuse source datafile location from Metadata File**: Select this option, or disable it and specify a new source datafile location to use for the unplugged PDB when it is plugged into the CDB.

   Click the right arrow button to go to the Storage page.

4. On the Storage page, select the type of location where you want to store the datafiles for the PDB:

   - If the target CDB (into which you are plugging the unplugged PDB) is enabled with Oracle Managed Files and you want to use the same, then select **Use Oracle Managed Files (OMF)**.

- If the target CDB does not use OMF, then specify a datafile location or accept the default specified in the **Datafile Location** field.

5. Also on the Storage page, choose whether or not to enable unlimited storage for the datafiles.

   If you do not enable unlimited storage, then you must specify values for these fields:

   - **Max Size**: The amount of storage that can be used by all tablespaces that belong to the PDB.
   - **Max Shared Temp Size**: The amount of storage in the default temporary tablespace shared by all PDBs that can be used by sessions connected to the PDB.

   If no values are specified for these fields, then their values will be set to unlimited when the PDB is plugged in.

6. If the CDB has a current active resource plan, the Resource Limits page appears.

   Supply values for these fields, or keep the default values for the PDB you are plugging in:

   - **Shares**: Assign a new share value for this PDB, or keep the current value.
   - **CPU Utilization Limit (%)**: Assign a new CPU utilization percentage value for this PDB, or keep the current value.
   - **Parallel Server Limit (%)**: Assign a new parallel server utilization percentage value for this PDB, or keep the current value.

7. Click **OK**.

   The unplugged PDB is plugged into the CDB and is opened in read/write mode. EM Express adds the PDB to the list of containers that appears in the Containers section on the Containers page.

After configuring an HTTPS port for EM Express for this PDB, the PDB can be managed using EM Express. Administrative users who have been granted the `EM_EXPRESS_ALL` role in a PDB can use EM Express to manage the PDB.

See Also:

- "Configuring the HTTPS Port for EM Express" for information on configuring an HTTPS port for EM Express for a PDB
- "Unplugging a PDB" for information on unplugging a PDB from a CDB

Removing PDBs Using EM Express

This section provides information about unplugging PDBs and deleting PDBs using EM Express. It includes the following topics:

- Unplugging a PDB
- Dropping a PDB

Unplugging a PDB Using EM Express

This section provides information about unplugging a PDB using EM Express.

Note:

Before you unplug a PDB from its CDB:

- The PDB that you want to unplug must have been opened at least once.
- The target host user must be the owner of the Oracle home that the CDB (that contains the PDB you want to unplug) belongs to.

To unplug a PDB from its CDB:

1. In EM Express, navigate to the Containers page for the CDB in which you want to unplug a PDB.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, click the PDB that you want to unplug, and then choose **Unplug** from the **Actions** menu. The Unplug wizard appears and advises you that as part of the unplug operation, the PDB is being closed with the Immediate option.

3. Click **OK**.

   The PDB is closed and unplugged. The Confirmation box appears and shows you the path to the metadata XML file created by the unplug operation. EM Express removes the PDB from the list of containers that appears in the Containers section on the Containers page.

Note:

If you decide later to plug the unplugged PDB into this CDB or a different CDB, you must know the location of the metadata XML file for the PDB.

See Also:

- "Plugging in an Unplugged PDB" for information on plugging an unplugged PDB into a CDB
- "Dropping a PDB"
- "Opening a PDB"

Dropping a PDB Using EM Express

This section provides information about dropping(permanently deleting) a PDB from a CDB using EM Express.

Note:

Before you drop a PDB from its CDB:

- The PDB that you want to drop must have been opened at least once.
- The target host user must be the owner of the Oracle home that the CDB (that contains the PDB you want to drop) belongs to.

To drop a PDB from its CDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the PDB that you want to drop.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, click the PDB that you want to drop, and then choose **Drop** from the **Actions** menu. The Drop wizard appears and advises you that the drop operation drops the resource plan directive associated with the PDB in the current active resource plan, and that the PDB is being closed with the Immediate option. By default, the datafiles for the PDB are also dropped.

3. Click **OK**.

   The PDB is dropped. The Confirmation box appears and advises you that the PDB was dropped successfully. EM Express removes the PDB from the list of containers that appears in the Containers section on the Containers page.

See Also:

- "Unplugging a PDB" for information on unplugging a PDB from its CDB
- "Opening a PDB"

Opening PDBs Using EM Express

This section provides information about opening one or all of the PDBs in a CDB using EM Express. It includes the following topics:

- Opening a PDB
- Opening All the PDBs in a CDB

Opening a PDB Using EM Express

This section provides information about opening a PDB in a CDB using EM Express.

Note:

Before you open a PDB:

- The PDB that you want to open must exist in the CDB.
- You must have the SYSDBA, SYSOPER, SYSBACKUP, or SYSDG administrative privilege, and the privilege must be either commonly granted or locally granted in the PDB. You must exercise the privilege using AS SYSDBA, AS SYSOPER, AS SYSBACKUP, or AS SYSDG, respectively, at connect time.

To open a PDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the PDB that you want to open.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, click the PDB that you want to open, and then choose **Open** from the **Actions** menu. The Open PDB wizard appears and asks you to specify the open mode for the PDB (**Read Write**, **Read Only**, or **Migrate**), and whether you want the PDB opened in restricted mode.

   See Also:

   Oracle Database Administrator's Guide for an introduction to the open modes of PDBs

   A PDB open in unrestricted mode allows access to all users, but a PDB open in restricted mode can be accessed only by a PDB administrator.

   Choose the open mode for the PDB and whether it should be opened in restricted mode or not.

   Note:

   If you choose to open a the PDB s that is already open, it will be closed with immediate mode, and then reopened in the open mode and restricted mode that you specified in the Open PDB wizard.

   If a PDB is closed immediately, users currently connected to the PDB are disconnected, and active transactions are implicitly rolled back.

   See *Oracle Database Administrator's Guide* for more information about shutting down using immediate mode.

3. Click **OK**.

   The PDB is opened. The Confirmation box appears and advises you that the PDB was opened successfully. In the Containers section on the Containers page, the up arrow icon appears in the Open Mode column for the PDB.

See Also:

- "Opening All the PDBs in a CDB"
- "Closing a PDB"
- "Closing All the PDBs in a CDB"

Opening All the PDBs in a CDB Using EM Express

This section provides information about opening all the PDBs in a CDB using EM Express.

Note:

Before you open all the PDBs in a CDB:

- Make sure that you want to open all the PDBs in the CDB.
- You must have the SYSDBA, SYSOPER, SYSBACKUP, or SYSDG administrative privilege, and the privilege must be either commonly granted or locally granted in the PDBs. You must exercise the privilege using AS SYSDBA, AS SYSOPER, AS SYSBACKUP, or AS SYSDG, respectively, at connect time.

To open all the PDBs in a CDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the PDBs that you want to open.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, choose **Open All** from the **Actions** menu. The Open All PDBs wizard appears and asks you to specify the open mode to use for all the PDBs (**Read Write**, **Read Only**, or **Migrate**), and whether you want the PDBs opened in restricted mode.

   See Also:

   Oracle Database Administrator's Guide for an introduction to the open modes of PDBs

   A PDB open in unrestricted mode allows access to all users, but a PDB open in restricted mode can be accessed only by a PDB administrator.

   Choose the open mode for the PDBs and whether they should be opened in restricted mode or not.

Caution:

When you choose to open all the PDBs in a CDB, if any PDBs in the CDB are already open, they will be closed immediately, and then reopened in the open mode and restricted mode that you specified in the Open All PDBs wizard.

If a PDB is closed immediately, users currently connected to the PDB are disconnected, and active transactions are implicitly rolled back.

See *Oracle Database Administrator's Guide* for more information about shutting down using immediate mode.

3. Click **OK**.

   All the PDBs in the CDB are opened. The Confirmation box appears and advises you that all the PDBs were opened successfully. In the Containers section on the Containers page, the up arrow icon appears in the Open Mode column for all the PDBs, and if they were opened in restricted mode, a check mark appears in the Restricted column for each PDB.

See Also:

- "Opening a PDB"
- "Closing a PDB"
- "Closing All the PDBs in a CDB"

Closing PDBs Using EM Express

This section provides information about closing one or all of the PDBs in a CDB using EM Express. It includes the following topics:

- Closing a PDB
- Closing All the PDBs in a CDB

Closing a PDB Using EM Express

This section provides information about closing a PDB in a CDB using EM Express.

Note:

Before you close a PDB in a CDB:

- The PDB that you want to close must exist in the CDB and must be open.
- You must have the `SYSDBA`, `SYSOPER`, `SYSBACKUP`, or `SYSDG` administrative privilege, and the privilege must be either commonly granted or locally granted in the PDB. You must exercise the privilege using `AS SYSDBA`, `AS SYSOPER`, `AS SYSBACKUP`, or `AS SYSDG`, respectively, at connect time.

To close a PDB in a CDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the PDB that you want to close.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, click the PDB that you want to close, and choose **Close** from the **Actions** menu. The Close PDB wizard appears and advises you that the PDB will be closed with the Immediate option.

3. Click **OK**.

   The PDB is closed. The Confirmation box appears and advises you that the PDB was closed successfully. In the Containers section on the Containers page, the down arrow icon appears in the Open Mode column for the PDB.

See Also:

- "Opening a PDB"
- "Opening All the PDBs in a CDB"
- "Closing All the PDBs in a CDB"

Closing All the PDBs in a CDB Using EM Express

This section provides information about closing all the PDBs in a CDB using EM Express.

Note:

Before you close all the PDBs in a CDB:

- At least one of the PDBs in the CDB must be open.
- You must have the `SYSDBA`, `SYSOPER`, `SYSBACKUP`, or `SYSDG` administrative privilege, and the privilege must be either commonly granted or locally granted in the PDBs that are open. You must exercise the privilege using `AS SYSDBA`, `AS SYSOPER`, `AS SYSBACKUP`, or `AS SYSDG`, respectively, at connect time.

To close all the PDBs in a CDB:

1. In EM Express, navigate to the Containers page for the CDB that contains the PDB that you want to close.

   For more information, see "Accessing the Containers Page."

2. In the Containers section of the Containers page, choose **Close All** from the **Actions** menu. The Close All PDBs wizard appears and advises you that the PDBs will be closed with the Immediate option.

3. Click **OK**.

All the PDBs are closed. The Confirmation box appears and advises you that the PDBs were closed successfully. In the Containers section on the Containers page, the down arrow icon appears in the Open Mode column for the PDBs.

See Also:

- "Opening a PDB"
- "Opening All the PDBs in a CDB"
- "Closing a PDB"

12

Managing Oracle Database Software

This chapter describes how to keep your Oracle Database software up-to-date. It contains the following sections:

About Software Management and Patch Releases

Software management involves keeping your Oracle Database software up-to-date with the latest product fixes. When a product defect, or a **bug**, is discovered, a patch is created to fix the problem. A **patch** corrects a single defect in the installed software. Individual patches, also referred to as **interim patches**, are made available to customers who for business reasons cannot wait until the next patch set to receive the product fix.

Oracle periodically issues maintenance releases for its software, in the form of patch sets. A **patch set** is a collection of product fixes that have been released up to the time of the maintenance release. Patch sets are fully tested and integrated product fixes. All the product fixes in the patch set have been tested and are certified to work with each other.

Every patch or patch set is associated with a bug number for identification purposes. Patch sets are also associated with version numbers. For example, if you use Oracle Database 11*g* Release 1 (11.1.0.6), then an available patch set might be 11.1.0.7. The version number of the patched software does not change if an interim patch is applied.

Every patch has an associated README file that describes how it fixes the software. The README file also has instructions for applying the patch.

Every patch set is accompanied by a Patch Set Notes file that contains installation instructions and information about the product fixes contained within the patch set. When you apply a patch set to your Oracle software, you change the maintenance release number for your installed software. Applying a patch set affects the software residing in your Oracle home only, with no change to the data in the database.

You can use Oracle OPatch and Oracle Universal Installer (OUI) to apply patches and patch sets respectively. Alternatively, you can automate the application of these patches using Oracle Enterprise Manager Cloud Control. Use of these products is beyond the scope of this documentation.

See Also:

Oracle Universal Installer and OPatch User's Guide for Windows and UNIX

Upgrading a Database

Use Database Upgrade Assistant (DBUA) to upgrade an existing database to the current release of Oracle Database.

This section contains these topics:

- Overview of Database Upgrade Assistant
- Database Releases Supported by DBUA
- Upgrade Scenarios for Oracle Database
- Starting DBUA
- Upgrading a Database Using DBUA
- Upgrading a PDB

Overview of Database Upgrade Assistant

Database Upgrade Assistant (DBUA) guides you through the upgrade process and configures your database for the new release. DBUA automates the upgrade process and makes appropriate recommendations for configuration options such as tablespaces and online redo log files.

DBUA can be used to upgrade databases created using any edition of the Oracle Database software, including Express Edition (XE) databases.

Pre-Upgrade Checks

DBUA does not begin the upgrade until it completes all of the following pre-upgrade steps:

- Checks for any invalid user accounts or roles
- Checks for any invalid data types or invalid objects
- Checks for any desupported character sets
- Checks for adequate resources, including rollback segments, tablespaces, and free disk space
- Checks for any missing SQL scripts needed for the upgrade
- Disables Database Vault, if Database Vault is enabled

  When upgrading Oracle Database to Oracle Database 12*c*, if you have enabled Database Vault in your current (pre-Oracle Database 12*c*) Oracle home, then by default Database Vault is disabled in the new target Oracle home. Once the upgrade is complete, enable Database Vault again.

See Also:

Oracle Database Upgrade Guide for more information about upgrading databases that use Database Vault

Automated Upgrade Tasks

After completing the pre-upgrade steps, DBUA automatically performs the following tasks:

- Modifies or creates new required tablespaces
- Invokes the appropriate upgrade scripts

While the upgrade is running, DBUA shows the upgrade progress for each component. DBUA writes detailed trace and log files and produces a complete HTML report for later reference. To enhance security, DBUA automatically locks new user accounts in the upgraded database. DBUA then proceeds to create new configuration files (initialization parameter and listener files) in the new Oracle home.

Support for Oracle Real Application Clusters

DBUA is fully compliant with Oracle Real Application Clusters (Oracle RAC) environments. In Oracle RAC environments, DBUA upgrades all database and configuration files on all nodes in the cluster.

About Upgrading Oracle Automatic Storage Management

Oracle ASM Configuration Assistant (ASMCA) enables you to upgrade an existing Oracle ASM instance to the current software level. However, the recommended practice is to upgrade an Oracle ASM instance with Oracle Universal Installer (OUI). OUI automatically defaults to upgrade mode when it detects an Oracle ASM instance at a previous release level.

See Also:

Oracle Automatic Storage Management Administrator's Guide for information about upgrading Oracle ASM

Support for Silent Mode

DBUA supports a silent mode of operation in which no user interface is presented to the user. Silent mode enables you to use a single statement for the upgrade.

Database Releases Supported by DBUA

If the release number of your current Oracle Database appears in the following list, you can use DBUA to perform a direct upgrade of your database to Oracle Database 12*c* Release 1 (12.1):

- 11.2.0.2 or later
- 11.1.0.7
- 10.2.0.4 or earlier

If your current Oracle Database is a release earlier than 10.2.0.4, or is release 11.2.0.1, then directly upgrading your current release of Oracle Database to the latest release is

not supported. In this case, you must upgrade to an intermediate release before upgrading to the new Oracle Database 12*c* release.

See Also:

Oracle Database Upgrade Guide for more information on upgrading your Oracle Database if a direct upgrade to the new Oracle Database 12*c* release is not supported.

Upgrade Scenarios for Oracle Database

The multitenant architecture introduced in Oracle Database 12*c* enables an Oracle database to function as a multitenant container database (CDB) that includes zero, one, or many customer-created pluggable databases (PDBs). A PDB is a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a non-CDB. All Oracle databases before Oracle Database 12*c* were non-CDBs.

With the introduction of CDBs and PDBs in Oracle Database 12*c*, more upgrade scenarios are available than in previous versions of Oracle Database.

This chapter includes information about these upgrade scenarios:

- Upgrading a pre-Oracle Database 12*c* Release 1 (12.1.0.2) non-CDB to an Oracle Database 12*c* Release 1 (12.1.0.2) non-CDB

  See "Upgrading a Database Using DBUA" for the steps to follow for this upgrade scenario.

- Upgrading a pre-Oracle Database 12*c* Release 1 (12.1.0.2) PDB to an Oracle Database 12*c* Release 1 (12.1.0.2) PDB

  See "Upgrading a PDB" for more information about this upgrade scenario.

See Also:

- *Oracle Database Concepts* for an overview of the multitenant architecture
- *Oracle Database Administrator's Guide* for complete information about creating and configuring a CDB
- *Oracle Database Upgrade Guide* for information about other Oracle Database upgrade scenarios

Starting DBUA

If you install the Oracle Database software only and specify that you are upgrading an existing database to the new Oracle Database release, then DBUA is launched automatically after the software installation. You can then continue as described in "Upgrading a Database Using DBUA."

If you perform a software-only installation and do not upgrade your database at that time, then you can do so later by launching DBUA.

Be aware of the following before you begin using DBUA:

- When upgrading a non-Oracle RAC database, you must run Oracle Net Configuration Assistant (NETCA) before running DBUA. When upgrading an

Oracle RAC database, as part of the Oracle Clusterware upgrade, Oracle Universal Installer (OUI) automatically runs NETCA to upgrade the network listener. Therefore, do not separately run NETCA.

- It is not possible to upgrade a database with DBUA when the source and target Oracle homes are owned by different users.
- If you stop the upgrade, but do not restore the database, then you cannot restart DBUA until you start the database instance in `UPGRADE` mode using the Oracle Database 12*c* server. You cannot go back to the original Oracle Database server version unless you restore your database. Instead, you must continue with a manual (command line) upgrade.
- If you restore your database manually (not using DBUA), then remove the following file from the Oracle Database 12*c* home directory before starting DBUA:

  ```
  $ ORACLE_HOME/cfgtoollogs/dbua/logs/Welcome_SID.txt
  ```

  The presence of this file indicates to DBUA that this is a re-run operation.

To start DBUA on Microsoft Windows:

1. Configure the operating system environment variables, as described in "Configuring the Operating System Environment Variables ."
2. Click **Start**, then select **Programs** (or **All Programs**)
3. Select **Oracle -** ***HOME_NAME***
4. Select **Configuration and Migration Tools**
5. Select **Database Upgrade Assistant**

   The DBUA Select Operation page appears.

To start DBUA on any supported platform:

1. Open a command window.
2. Configure the operating system environment variables, as described in "Configuring the Operating System Environment Variables ."
3. Enter the following command:

   ```
   dbua
   ```

   The Database Upgrade Assistant: Select Operation page appears.

Note:

The `dbua` executable is typically found in your *`Oracle_home`*`/bin` directory.

See Also:

- *Oracle Database Upgrade Guide* for more information about running Net Configuration Assistant prior to running DBUA
- *Oracle Database Upgrade Guide* for more information about manually upgrading a database
- *Oracle Database Upgrade Guide* for more information about DBUA, the DBUA pages, and other DBUA operations not described in this chapter

Upgrading a Database Using DBUA

Complete the following steps to upgrade a database using DBUA. If you need help at any page or want to consult more documentation about DBUA, then click the **Help** button to access the online Help.

To upgrade a database using DBUA:

1. Start DBUA. See "Starting DBUA."
2. At the Select Operation page of DBUA, choose **Upgrade Oracle Database**. Then, click **Next**.

   The Select Database page appears.
3. On the Select Database page, choose the database you want to upgrade to Oracle Database 12c Release 1 (12.1.0.2) from the **Source Database Oracle Home** list. DBUA fills in information about the database. DBUA fills in the information for the release and selects the Oracle database associated with the selected Oracle home.

   If the source database does not have operating system authentication, then DBUA prompts you for the user name and password of an account with `SYSDBA` privilege for the database.

   You can select only one database at a time. If the database that you want to upgrade does not appear in the list, then make sure an entry with the database name exists in the `oratab` file in the `etc` directory.

   If you run DBUA from a user account that does not have `SYSDBA` privileges, then enter the user name and password for an account with `SYSDBA` privilege for the selected database.

   Click **Next**.
4. If the selected database is a multitenant container database (CDB), then DBUA displays the Pluggable Databases page. The Pluggable Databases page lists the pluggable databases (PDBs) contained in the CDB, which will be upgraded along with the selected CDB.

   Click **Next**.
5. DBUA analyzes the database, performing pre-upgrade checks and displaying warnings as necessary. Examples of DBUA database checks include:

   - Empty database recycle bin

- Invalid objects
- Deprecated and desupported initialization parameters
- Time zone data file version

When DBUA finishes its analysis, the Prerequisite Checks page appears. The analysis takes several minutes to complete.

6. The Prerequisite Checks page shows the validation and results, the severity, whether the result is fixable if there is a problem, and the action you can take. When you select the validation result, the **Fixable** column displays whether the result can be fixed or not. The **Action** drop-down list shows actions you can take. For example, select **Fix** for DBUA to run a script or command to fix the problem.

   When you select a result in the **Validation** column, DBUA displays information about the result in the bottom area of the screen.

   - Click the **more details** link in the text information area. The Validation Details box appears with more information.
   - If there are validation errors or warnings and these are fixable, you can select an Action to take.

   Click **Next**.

   The Upgrade Options page appears with the Upgrade Options tab selected.

7. The Upgrade Options page with the Upgrade Options tab selected provides the following options:

   Select Upgrade Parallelism

   The Select Upgrade Parallelism section enables the degree of parallelism for the upgrade process. This option reduces the time needed to perform the upgrade, based on the number of CPUs available to handle the running of scripts and processes simultaneously.

   By default, DBUA sets Upgrade Parallelism to the number of CPUs or 2 if the number of CPUs is less than 4. You can adjust this default value by selecting a new value from the **Select Upgrade Parallelism** list.

   Recompile Invalid Objects During Post-Upgrade

   Select **Recompile Invalid Objects During Post-Upgrade** if you want DBUA to recompile all invalid PL/SQL modules after the upgrade is complete. Specify the parallelism for the recompilation of invalid objects during post upgrade. DBUA provides a recommended degree of Recompilation Parallelism, which it sets to one less than the number of CPUs you have available. Taking advantage of parallism can significantly reduce the upgrade time. If you do not have DBUA recompile invalid objects in its post-upgrade phase, then you must manually recompile invalid objects after the database is upgraded.

   Upgrade Timezone Data

   Select **Upgrade Timezone Data** for DBUA to update the time zone data file for this release. If you do not select this option, then you must update the time zone configuration file manually after the upgrade.

   Gather Statistics Before Upgrade

   Select **Gather Statistics Before Upgrade** to reduce the overall time for the upgrade process by gathering statistics before upgrading.

Set User Tablespace to Read Only During the Upgrade

Disable **Set User Tablespace to Read Only During the Upgrade** if you are upgrading a database in which you must transport tablespaces. Transportable tablespaces must have writable file headers.

Diagnostic Destination

The **Diagnostic Destination** field specifies the location for output that DBUA creates for diagnostics. You can accept the default, enter a full path into the field, or click **Browse** to navigate to a location.

Audit File Destination

The Audit File Destination field specifies the location for DBUA to save audit files. Accept the default, enter a full path into the field, or click **Browse** to navigate to a location.

Optionally, click the **Custom Scripts** tab to specify custom SQL scripts that you would like to run before and after the upgrade

The Upgrade Options page appears with the Custom Scripts tab selected.

You may also click **Next** without using the Custom Scripts option.

8. The Custom Scripts tab in the Upgrade Options page allows you to optionally run custom SQL scripts. If you want to run a script before the upgrade, click **Browse** for the **Before Upgrade** field to browse to the location of the custom SQL script you would like to run before the upgrade. If you want to run a script after the upgrade, click **Browse** for the **After Upgrade** field to browse to the location of the custom SQL script you would like to run after the upgrade.

   You can specify either one or both, or leave the fields blank to skip this tab.

   Click **Next**.

   The Management Options page appears.

9. In the Management Options page, select an option:

   - **Configure Enterprise Manager (EM) Database Express**

     You can enter the EM Express port number, for example, 5500.

   - **Register with Enterprise Manager (EM) Cloud Control**

     Registering with Oracle Enterprise Manager Cloud Control adds the database and its related entities, such as listener, Oracle ASM disk groups, and Oracle Clusterware as managed targets.

     If you select this option, then you must provide information in the following fields:

     - OMS Host
     - OMS Port
     - EM Admin Username
     - EM Admin Password
     - DBSNMP Password

   Click **Next**.

If you are upgrading a single-instance database (the assumption for readers of this manual) or Oracle Express Edition (XE), the Move Database Files page appears.

10. On the Move Database Files page, select an option:

    - **Move Database Files as Part of Upgrade**
    - **Move Fast Recovery Area as Part of Upgrade**

    Note:

    The fast recovery area is an Oracle managed disk location used for storing backup and recovery related files. Oracle strongly recommends configuring a fast recovery area, because it significantly enhances speed, reliability, and manageability of the database recovery process. The location of the fast recovery area is also used by Oracle Enterprise Manager if you enable local management on the Management Options screen.

    If you are upgrading an Oracle Express Edition database, the Move Database Files page also includes the Rename Database section with the **Global Database Name** and **SID** fields. You must provide values for these fields. The rest of the Move Database Files page options are the same as for Oracle Database.

11. If you choose **Move Database Files as Part of Upgrade**, then you must also configure **Storage Type** for the database files.

    In the **Storage Type** list, select **File System** or **Oracle ASM**.

    - If you select **File System**, your database files are moved to the host file system.
    - If you select **Oracle Automatic Storage Management (Oracle ASM)**, your database files are moved to Oracle ASM storage, which must currently exist on your system. If you do not have an Oracle ASM instance, then you can create one using Automatic Storage Management Configuration Assistant (ASMCA) from the Oracle Grid Infrastructure home and then restart DBUA.

    See Also:

    - *Oracle Database Upgrade Guide* for more information about moving database files
    - *Oracle Grid Infrastructure Installation Guide* for information about installing and configuring Oracle ASM
    - *Oracle Automatic Storage Management Administrator's Guide* for information about managing Oracle ASM instances with ASMCA

    - You can choose either **Use Common Location for All Database Files** or **Use a Mapping File to Specify Location of Database Files**. Specify the location in the file location field or browse to the location.
    - If you choose **Oracle Managed Files**, then click **Multiplex Redo Logs and Control Files**. The Multiplex Redo Logs and Control Files dialog box appears with location fields. Enter locations for online redo logs and control files to be

written. Use multiple locations spread across different disks to provide greater fault tolerance.

12. If you choose **Move Fast Recovery Area as Part of Upgrade**, then you must also configure the storage type and location for the fast recovery area and specify the size to be allocated as described in this step.

    - The Move Database Files and Move Fast Recovery Area options are independent of each other. For example, you can choose to move database files to Oracle ASM and leave the fast recovery area on the file system.
    - When you choose to move the fast recovery area, DBUA does not physically move existing archived redo logs to a new location. Instead, DBUA sets the `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` initialization parameters to the new location and new size when the database is started from the new Oracle home.
    - If an Oracle Express Edition database is being upgraded to Oracle Enterprise Edition, then you must configure a fast recovery area. If a fast recovery area is currently configured, then the current settings are retained but the page displays to enable you to override these values.

    Fast Recovery Area Storage Type

    In the **Storage Type** list, select **File System** or **ASM**.

    If you select **File System**, your fast recovery area will be on the host file system.

    If you select **Oracle Automatic Storage Management (Oracle ASM)**, your fast recovery area will be on Oracle ASM storage, which must currently exist on your system. If you do not have an Oracle ASM instance, then you can create one using Automatic Storage Management Configuration Assistant (ASMCA) from the Oracle Grid Infrastructure home and then restart DBUA.

    Fast Recovery Area Location

    Browse to the location on the host file system, or on Oracle ASM storage.

    Fast Recovery Area Size

    Specify the size to allocate for the fast recovery area. The default is 1024 MB.

    Click **Next**. The Network Configuration page appears with the Listener Selection tab showing.

13. The **Listener Selection** tab on the Network Configuration page shows a table with: **Name**, **Port**, **Oracle Home**, **Status**, and **Migrate** columns. To the left of the listener name is a box for selecting the listener.

    Select one or more listeners from the source Oracle home to be migrated to the new upgraded Oracle home.

    a. DBUA adds the selected listener to the `listener.ora`. of the target Oracle home and starts it.

    b. DBUA removes the entry of the upgraded database from the old (source) `listener.ora` file.

    c. DBUA reloads the `listener.ora` file in both the source and target Oracle Database environments.

Note:

If there are other databases registered with the same listener, then their new client connection requests may be affected during listener migration.

Select **Create a New Listener** to create a new listener. Provide the name and port number.

Click **Next**.

DBUA displays the Recovery Options page. DBUA performs the listener migration during the pre-upgrade steps.

14. In the Recovery Options Configuration page, select the recovery method to use if the upgrade process encounters a problem.

    The following recovery options and configurations are available:

    Use RMAN Backup

    If you select **Create RMAN Backup before Upgrade**, then enter the full path for a location for the backup in the **Backup Location** field.

    If you select **Use Latest Available RMAN Backup** but do not choose to create an RMAN backup before upgrading, then DBUA displays the time stamp for the latest RMAN backup that exists. You can click **Restore Script** next to the time stamp to select an existing script to run for restoring this backup.

    Use Flashback and Guaranteed Restore Point

    Select **Create a New Guaranteed Restore Point** to have DBUA create a restore point before DBUA enters the upgrade process.

    If you previously enabled Flashback Database and configured a fast recovery area with a flashback retention target, then you can select **Use Available Guaranteed Restore Point** and pick the named SCN from the drop-down list. Your current settings for the restore point are retained. DBUA displays a page to permit you to override these values if needed.

    I have my own backup and restore strategy

    Click this option only if you used your own backup procedure to back up the database. In this case, Restore restores only the original database settings. To restore the database itself, you must restore the backup you created with your own backup utilities.

    Note:

    The database you are upgrading must be release 11.1.0.7 or later in order to take advantage of Flashback and Guaranteed Restore Point, and this must be enabled in the source database.

    See Also:

    Oracle Database Backup and Recovery User's Guide for more information on using Flashback Database and guaranteed restore points

    Click **Next**.

The Summary page appears.

15. The Summary page shows the following information about the upgrade before it starts:

    - Source Database
    - Target Database
    - Pluggable Databases
    - Pre-Upgrade Checks
    - Initialization Parameter changes
    - Timezone Upgrade

    See Also:

    Oracle Database Upgrade Guide for information about setting the `COMPATIBLE` initialization parameter after the upgrade

    Check all of the details. Then click **Back** or **Finish** as follows:

    - Click **Back** if anything is incorrect until you reach the page where you can correct it.
    - Click **Finish** if everything is correct.

    The Progress page appears and DBUA begins the upgrade.

16. The Progress page shows a table with the steps being performed, the time duration, and the status as the upgrade proceeds. DBUA provides a **Stop** button in case you must cancel the upgrade at this point.

    When the upgrade has progressed through finishing the upgrade, the Progress page marks the status Finished. You can click **Activity Log**, **Alert Log**, and **Upgrade Results** to view more information.

    After the upgrade has completed, the Upgrade Results page appears.

17. The Upgrade Results page displays a description of the original database and the upgraded database and shows the changes made to the initialization parameters. The page also shows the directory where various log files are stored after the upgrade, and PDBs (if you upgraded a CDB with PDBs). Scroll down to view more details about pre-upgrade checks.

18. Optionally, you can examine the log files to obtain more details about the upgrade process. The DBUA log files are located under the `/oracle_base/cfgtoollogs/dbua/logs` directory.

    Note:

    An HTML version of the Upgrade Results is also saved in the log files directory. You can click the links in this HTML page to view the log pages in your web browser.

If you are satisfied with the upgrade results, then click **Close** to quit DBUA and use your newly upgraded database.

See Also:

Oracle Database Upgrade Guide for information about additional tasks that should be completed after upgrading your database

Note:

To prevent unauthorized use of the database, Oracle recommends that you change all user passwords immediately after you upgrade your database.

If the default security settings for Oracle Database 12c are in place, then passwords must be at least eight characters, and passwords such as `welcome` and `oracle` are not allowed. See *Oracle Database Security Guide* for information about configuring authentication.

Upgrading a PDB

A CDB can contain zero, one, or more PDBs. You can upgrade one PDB without upgrading the whole CDB. For example, you can unplug a PDB from a release 12.1.0.1 CDB, plug it into a release 12.1.0.2 CDB, and then upgrade that PDB to release 12.1.0.2. You cannot use DBUA to upgrade a PDB, therefore you manually upgrade the PDB.

See:

Oracle Database Upgrade Guide for the steps to follow to upgrade a PDB

Removing Oracle Database Software

If you want to remove an Oracle software installation, you can use the deinstallation tool to completely uninstall the software from your computer.

Starting with Oracle Database 12*c*, the deinstallation tool is integrated with the database installation media. You can run the deinstallation tool using the `runInstaller` command with the `-deinstall` and `-home` options from the base directory of the Oracle Database, Oracle Database Client or Oracle Grid Infrastructure installation media.

The deinstallation Tool (`deinstall`) is also available in Oracle home directories after installation as `$ORACLE_HOME/deinstall/deinstall`.

The default method for running the deinstallation tool is from the `deinstall` directory in the Oracle home as the installation owner:

To remove an Oracle home from your computer:

1. Execute the deinstallation tool using these commands:

```
$ cd $ORACLE_HOME
$ $ORACLE_HOME/deinstall/deinstall
```

Note:

Do not shut down the Oracle database or stop any database processes prior to running the deinstallation tool.

2. Enter the specified information at the prompts, or just press Enter to accept the default values.
3. When prompted, follow the directions to run the root script. You must be the `root` user to do so.

See Also:

Oracle Database Installation Guide for your platform for more information about the deinstallation tool

Managing Oracle Software: Oracle by Example Series

Oracle By Example (OBE) has a series on the *Oracle Database 2 Day DBA* guide. This OBE steps you through the tasks in this chapter, and includes annotated screenshots.

To view the Managing Oracle Database Software OBE, enter the following URL in your web browser:

```
https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:
24:P24_CONTENT_ID,P24_PREV_PAGE:6291,1
```

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W