Oracle®

High Availability Architecture and Best Practices

10*g* Release 1 (10.1)

Part No. B10726-01

December 2003

ORACLE

Oracle High Availability Architecture and Best Practices, 10*g* Release 1 (10.1)

Part No. B10726-01

Primary Author: Cathy Baird

Contributors: David Austin, Andrew Babb, Mark Bauer, Ruth Baylis, Tammy Bednar, Pradeep Bhat, Donna Cooksey, Ray Dutcher, Jackie Gosselin, Mike Hallas, Daniela Hansell, Wei Hu, Susan Kornberg, Jeff Levinger, Diana Lorentz, Roderick Manalac, Ashish Ray, Antonio Romero, Vivian Schupmann, Deborah Steiner, Ingrid Stuart, Bob Thome, Lawrence To, Paul Tsien, Douglas Utzig, Jim Viscusi, Shari Yamaguchi

Graphic Artist: Valarie Moore

Contents

Send Us Your Comments xv

Preface xvii

Audience xvii
Organization xviii
Related Documentation xx
Conventions xx
Documentation Accessibility xxiii

Part I Getting Started

1 Overview of High Availability

Introduction to High Availability 1-2
What is Availability? 1-2
Importance of Availability 1-3
Causes of Downtime 1-4
What Does This Book Contain? 1-4
Who Should Read This Book? 1-5

2 Determining Your High Availability Requirements

Why It Is Important to Determine High Availability Requirements 2-2
Analysis Framework for Determining High Availability Requirements 2-3
Business Impact Analysis 2-3
Cost of Downtime 2-3

Recovery Time Objective 2-4
Recovery Point Objective 2-5
Choosing a High Availability Architecture 2-5
HA Systems Capabilities 2-7
Business Performance, Budget and Growth Plans 2-8
High Availability Best Practices 2-9

Part II Oracle Database High Availability Features, Architectures, and Policies

3 Oracle Database High Availability Features

Oracle Real Application Clusters 3-2
Oracle Data Guard 3-2
Oracle Streams 3-4
Online Reorganization 3-5
Transportable Tablespaces 3-5
Automatic Storage Management 3-6
Flashback Technology 3-7
Oracle Flashback Query 3-7
Oracle Flashback Version Query 3-8
Oracle Flashback Transaction Query 3-8
Oracle Flashback Table 3-8
Oracle Flashback Drop 3-8
Oracle Flashback Database 3-8
Dynamic Reconfiguration 3-8
Oracle Fail Safe 3-9
Recovery Manager 3-10
Flash Recovery Area 3-11
Hardware Assisted Resilient Data (HARD) Initiative 3-11

4 High Availability Architectures

Oracle Database High Availability Architectures 4-2
"Database Only" Architecture 4-3
"RAC Only" Architecture 4-5
"Data Guard Only" Architecture 4-7

Maximum Availability Architecture 4-9
Streams Architecture 4-10
Choosing the Correct HA Architecture 4-12
Assessing Other Architectures 4-15

5 Operational Policies for High Availability

Introduction to Operational Policies for High Availability 5-2
Service Level Management for High Availability 5-2
Planning Capacity to Promote High Availability 5-4
Change Management for High Availability 5-4
Backup and Recovery Planning for High Availability 5-7
Disaster Recovery Planning 5-8
Planning Scheduled Outages 5-9
Staff Training for High Availability 5-11
Documentation as a Means of Maintaining High Availability 5-11
Physical Security Policies and Procedures for High Availability 5-13

Part III Configuring a Highly Available Oracle Environment

6 System and Network Configuration

Overview of System Configuration Recommendations 6-2
Recommendations for Configuring Storage 6-2
Ensure That All Hardware Components Are Fully Redundant and Fault-Tolerant 6-3
Use an Array That Can Be Serviced Online 6-3
Mirror and Stripe for Protection and Performance 6-3
Load-Balance Across All Physical Interfaces 6-4
Create Independent Storage Areas 6-4
Storage Recommendations for Specific HA Architectures 6-5
Define ASM Disk and Failure Groups Properly 6-6
Use HARD-Compliant Storage for the Greatest Protection Against Data Corruption 6-8
Storage Recommendation for RAC 6-9
Protect the Oracle Cluster Registry and Voting Disk From Media Failure 6-9
Recommendations for Configuring Server Hardware 6-9
Server Hardware Recommendations for All Architectures 6-9

Use Fewer, Faster, and Denser Components 6-10
Use Redundant Hardware Components 6-10
Use Systems That Can Detect and Isolate Failures 6-10
Protect the Boot Disk With a Backup Copy 6-10
Server Hardware Recommendations for RAC 6-10
Use a Supported Cluster System to Run RAC 6-11
Choose the Proper Cluster Interconnect 6-11
Server Hardware Recommendations for Data Guard 6-11
Use Identical Hardware for Every Machine at Both Sites 6-11
Recommendations for Configuring Server Software 6-11
Server Software Recommendations for All Architectures 6-12
Use the Same OS Version, Patch Level, Single Patches, and Driver Versions 6-12
Use an Operating System That is Fault-Tolerant to Hardware Failures 6-12
Configure Swap Partititions Appropriately 6-12
Set Operating System Parameters to Enable Future Growth 6-13
Use Logging or Journal File Systems 6-13
Mirror Disks That Contain Oracle and Application Software 6-13
Server Software Recommendations for RAC 6-13
Use Supported Clustering Software 6-13
Use Network Time Protocol (NTP) On All Cluster Nodes 6-14
Recommendations for Configuring the Network 6-14
Network Configuration Best Practices for All Architectures 6-14
Ensure That All Network Components Are Redundant 6-14
Use Load Balancers to Distribute Incoming Requests 6-16
Network Configuration Best Practices for RAC 6-17
Classify Network Interfaces Using the Oracle Interface Configuration Tool 6-17
Network Configuration Best Practices for Data Guard 6-17
Configure System TCP Parameters Appropriately 6-17
Use WAN Traffic Managers to Provide Site Failover Capabilities 6-18

7 Oracle Configuration Best Practices

Configuration Best Practices for the Database 7-2
Use Two Control Files 7-2
Set CONTROL_FILE_RECORD_KEEP_TIME Large Enough 7-3
Configure the Size of Redo Log Files and Groups Appropriately 7-3

Multiplex Online Redo Log Files 7-3
Enable ARCHIVELOG Mode 7-4
Enable Block Checksums 7-4
Enable Database Block Checking 7-5
Log Checkpoints to the Alert Log 7-5
Use Fast-Start Checkpointing to Control Instance Recovery Time 7-5
Capture Performance Statistics About Timing 7-7
Use Automatic Undo Management 7-7
Use Locally Managed Tablespaces 7-9
Use Automatic Segment Space Management 7-9
Use Temporary Tablespaces and Specify a Default Temporary Tablespace 7-9
Use Resumable Space Allocation 7-9
Use a Flash Recovery Area 7-10
Enable Flashback Database 7-10
Set Up and Follow Security Best Practices 7-11
Use the Database Resource Manager 7-11
Use a Server Parameter File 7-12
Configuration Best Practices for Real Application Clusters 7-12
Register All Instances with Remote Listeners 7-13
Do Not Set CLUSTER_INTERCONNECTS Unless Required for Scalability 7-13
Configuration Best Practices for Data Guard 7-13
Use a Simple, Robust Archiving Strategy and Configuration 7-16
Use Multiplexed Standby Redo Logs and Configure Size Appropriately 7-19
Enable FORCE LOGGING Mode 7-19
Use Real Time Apply 7-20
Configure the Database and Listener for Dynamic Service Registration 7-20
Tune the Network in a WAN Environment 7-22
Determine the Data Protection Mode 7-22
Determining the Protection Mode 7-23
Changing the Data Protection Mode 7-24
Conduct a Performance Assessment with the Proposed Network Configuration 7-25
Use a LAN or MAN for Maximum Availability or Maximum Protection Modes 7-26
Set SYNC=NOPARALLEL / PARALLEL Appropriately 7-27
Use ARCH for the Greatest Performance Throughput 7-28
Use the ASYNC Attribute with a 50 MB Buffer for Maximum Performance Mode 7-29

Evaluate SSH Port Forwarding with Compression 7-30
Set LOG_ARCHIVE_LOCAL_FIRST to TRUE 7-31
Provide Secure Transmission of Redo Data 7-31
Set DB_UNIQUE_NAME 7-32
Set LOG_ARCHIVE_CONFIG Correctly 7-32
Recommendations for the Physical Standby Database Only 7-32
Tune Media Recovery Performance 7-32
Recommendations for the Logical Standby Database Only 7-33
Use Supplemental Logging and Primary Key Constraints 7-33
Set the MAX_SERVERS Initialization Parameter 7-33
Increase the PARALLEL_MAX_SERVERS Initialization Parameter 7-34
Set the TRANSACTION_CONSISTENCY Initialization Parameter 7-34
Skip SQL Apply for Unnecessary Objects 7-35
Configuration Best Practices for MAA 7-35
Configure Multiple Standby Instances 7-35
Configure Connect-Time Failover for Network Service Descriptors 7-36
Recommendations for Backup and Recovery 7-36
Use Recovery Manager to Back Up Database Files 7-37
Understand When to Use Backups 7-37
Perform Regular Backups 7-38
Initial Data Guard Environment Set-Up 7-38
Recovering from Data Failures Using File or Block Media Recovery 7-38
Double Failure Resolution 7-38
Long-Term Backups 7-39
Use an RMAN Recovery Catalog 7-39
Use the Autobackup Feature for the Control File and SPFILE 7-39
Use Incrementally Updated Backups to Reduce Restoration Time 7-39
Enable Change Tracking to Reduce Backup Time 7-40
Create Database Backups on Disk in the Flash Recovery Area 7-40
Create Tape Backups from the Flash Recovery Area 7-40
Determine Retention Policy and Backup Frequency 7-40
Configure the Size of the Flash Recovery Area Properly 7-41
In a Data Guard Environment, Back Up to the Flash Recovery Area on All Sites 7-41
During Backups, Use the Target Database Control File as the RMAN Repository 7-42
Regularly Check Database Files for Corruption 7-43

Periodically Test Recovery Procedures 7-43
Back Up the OCR to Tape or Offsite 7-43
Recommendations for Fast Application Failover 7-44
Configure Connection Descriptors for All Possible Production Instances 7-45
Use RAC Availability Notifications and Events 7-47
Use Transparent Application Failover If RAC Notification Is Not Feasible 7-47
New Connections 7-48
Existing Connections 7-48
LOAD_BALANCE Parameter in the Connection Descriptor 7-49
FAILOVER Parameter in the Connection Descriptor 7-49
SERVICE_NAME Parameter in the Connection Descriptor 7-49
RETRIES Parameter in the Connection Descriptor 7-49
DELAY Parameter in the Connection Descriptor 7-49
Configure Services 7-50
Configure CRS for High Availability 7-50
Configure Service Callouts to Notify Middle-Tier Applications and Clients 7-50
Publish Standby or Nonproduction Services 7-51
Publish Production Services 7-51

Part IV Managing a Highly Available Oracle Environment

8 Using Oracle Enterprise Manager for Monitoring and Detection

Overview of Monitoring and Detection for High Availability 8-2
Using Enterprise Manager for System Monitoring 8-2
Set Up Default Notification Rules for Each System 8-5
Use Database Target Views to Monitor Health, Availability, and Performance 8-9
Use Event Notifications to React to Metric Changes 8-10
Use Events to Monitor Data Guard system Availability 8-11
Managing the HA Environment with Enterprise Manager 8-11
Check Enterprise Manager Policy Violations 8-12
Use Enterprise Manager to Manage Oracle Patches and Maintain System Baselines 8-12
Use Enterprise Manager to Manage Data Guard Targets 8-12
Highly Available Architectures for Enterprise Manager 8-13
Recommendations for an HA Architecture for Enterprise Manager 8-15
Protect the Repository and Processes As Well as the Configuration They Monitor . 8-15

Place the Management Repository in a RAC Instance and Use Data Guard 8-16
Configure At Least Two Management Service Processes and Load Balance Them.. 8-16
Consider Hosting Enterprise Manager on the Same Hardware as an HA System.... 8-16
Monitor the Network Bandwidth Between Processes and Agents 8-16
Unscheduled Outages for Enterprise Manager 8-17
Additional Enterprise Manager Configuration 8-18
Configure a Separate Listener for Enterprise Manager 8-19
Install the Management Repository Into an Existing Database 8-19

9 Recovering from Outages

Recovery Steps for Unscheduled Outages 9-2
Recovery Steps for Unscheduled Outages on the Primary Site 9-4
Recovery Steps for Unscheduled Outages on the Secondary Site 9-6
Recovery Steps for Scheduled Outages 9-7
Recovery Steps for Scheduled Outages on the Primary Site 9-9
Recovery Steps for Scheduled Outages on the Secondary Site 9-11
Preparing for Scheduled Secondary Site Maintenance 9-12

10 Detailed Recovery Steps

Summary of Recovery Operations 10-2
Complete or Partial Site Failover 10-3
Complete Site Failover 10-3
Partial Site Failover: Middle-Tier Applications Connect to a Remote Database Server .. 10-7
Database Failover 10-10
When to Use Data Guard Failover 10-10
When Not to Use Data Guard Failover 10-11
Data Guard Failover Using SQL*Plus 10-11
Physical Standby Failover Using SQL*Plus 10-11
Logical Standby Failover Using SQL*Plus 10-12
Database Switchover 10-12
When to Use Data Guard Switchover 10-13
When Not to Use Data Guard Switchover 10-13
Data Guard Switchover Using SQL*Plus 10-13
Physical Standby Switchover Using SQL*Plus 10-14
Logical Standby Switchover Using SQL*Plus 10-14

RAC Recovery 10-15
RAC Recovery for Unscheduled Outages 10-15
Automatic Instance Recovery for Failed Instances 10-15
Single Node Failure in Real Application Clusters 10-16
Multiple Node Failures in Real Application Clusters 10-16
Automatic Service Relocation 10-16
RAC Recovery for Scheduled Outages 10-17
Disabling CRS-Managed Resources 10-17
Planned Service Relocation 10-17
Apply Instance Failover 10-18
Performing an Apply Instance Failover Using SQL*Plus 10-19
Step 1: Ensure That the Chosen Standby Instance is Mounted 10-20
Step 2: Verify Oracle Net Connection to the Chosen Standby Host 10-20
Step 3: Start Recovery on the Chosen Standby Instance 10-20
Step 4: Copy Archived Redo Logs to the New Apply Host 10-21
Step 5: Verify the New Configuration 10-21
Recovery Solutions for Data Failures 10-22
Detecting and Recovering From Datafile Block Corruption 10-23
Detecting Datafile Block Corruption 10-23
Recovering From Datafile Block Corruption 10-24
Determine the Extent of the Corruption Problem 10-24
Replace or Move Away From Faulty Hardware 10-26
Determine Which Objects Are Affected 10-26
Decide Which Recovery Method to Use 10-26
Recovering From Media Failure 10-29
Determine the Extent of the Media Failure 10-29
Replace or Move Away From Faulty Hardware 10-30
Decide Which Recovery Action to Take 10-30
Recovery Methods for Data Failures 10-32
Use RMAN Datafile Media Recovery 10-33
Use RMAN Block Media Recovery 10-33
Re-Create Objects Manually 10-34
Use Data Guard to Recover From Data Failure 10-35
Recovering from User Error with Flashback Technology 10-35
Resolving Row and Transaction Inconsistencies 10-37

Flashback Query 10-38
Flashback Version Query 10-38
Flashback Transaction Query 10-39
Example: Using Flashback Technology to Investigate Salary Discrepancy 10-39
Resolving Table Inconsistencies 10-41
Flashback Table 10-41
Flashback Drop 10-42
Resolving Database-Wide Inconsistencies 10-42
Flashback Database 10-42
Using Flashback Database to Repair a Dropped Tablespace 10-44
RAC Rolling Upgrade 10-45
Applying a Patch with opatch 10-46
Rolling Back a Patch with opatch 10-47
Using opatch to List Installed Software Components and Patches 10-47
Recommended Practices for RAC Rolling Upgrades 10-48
Upgrade with Logical Standby Database 10-49
Online Object Reorganization 10-53
Online Table Reorganization 10-54
Online Index Reorganization 10-54
Online Tablespace Reorganization 10-54

11 Restoring Fault Tolerance

Restoring Full Tolerance 11-2
Restoring Failed Nodes or Instances in a RAC Cluster 11-2
Recovering Service Availability 11-4
Considerations for Client Connections After Restoring a RAC Instance 11-5
Restoring the Standby Database After a Failover 11-10
Restoring a Physical Standby Database After a Failover 11-10
Step 1P: Retrieve STANDBY_BECAME_PRIMARY_SCN 11-11
Step 2P: Flash Back the Previous Production Database 11-11
Step 3P: Mount New Standby Database From Previous Production Database 11-11
Step 4P: Archive to New Standby Database From New Production Database 11-12
Step 5P: Start Managed Recovery 11-12
Step 6P: Restart MRP After It Encounters the End-of-Redo Marker 11-12
Restoring a Logical Standby Database After a Failover 11-13

Step 1L: Retrieve END_PRIMARY_SCN 11-13
Step 2L: Flash Back the Previous Production Database 11-13
Step 3L: Open New Logical Standby Database and Start SQL Apply 11-13
Restoring Fault Tolerance after Secondary Site or Clusterwide Scheduled Outage 11-14
Step 1: Start the Standby Database 11-14
Step 2: Start Recovery 11-14
Step 3: Verify Log Transport Services on Production Database 11-14
Step 4: Verify that Recovery is Progressing on Standby Database 11-15
Step 5: Restore Production Database Protection Mode 11-15
Restoring Fault Tolerance after a Standby Database Data Failure 11-16
Step 1: Fix the Cause of the Outage 11-16
Step 2: Restore the Backup of Affected Datafiles 11-16
Step 3: Restore Required Archived Redo Log Files 11-16
Step 4: Start the Standby Database 11-17
Step 5: Start Recovery or Apply 11-17
Step 6: Verify Log Transport Services On the Production Database 11-17
Step 7: Verify that Recovery or Apply Is Progressing On the Standby Database 11-18
Step 8: Restore Production Database Protection Mode 11-18
Restoring Fault Tolerance After the Production Database Has Opened Resetlogs 11-18
Scenario 1: SCN on Standby is Behind Resetlogs SCN on Production 11-19
Scenario 2: SCN on Standby is Ahead of Resetlogs SCN on Production 11-19
Restoring Fault Tolerance after Dual Failures 11-20

A Hardware Assisted Resilient Data (HARD) Initiative

Preventing Data Corruptions with HARD-Compliant Storage A-1
Data Corruptions A-2
Types of Data Corruption Addressed by HARD A-2
Possible HARD Checks A-3

B Database SPFILE and Oracle Net Configuration File Samples

SPFILE Samples N-2
Oracle Net Configuration Files N-8
SQLNET.ORA File Example for All Hosts Using Dynamic Instance Registration N-8
LISTENER.ORA File Example for All Hosts Using Dynamic Instance Registration N-8

TNSNAMES.ORA File Example for All Hosts Using Dynamic Instance Registration N-9

Index

Send Us Your Comments

Oracle High Availability Architecture and Best Practices, 10*g* Release 1 (10.1)

Part No. B10726-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227 Attn: Server Technologies Documentation Manager
- Postal service:
  Oracle Corporation
  Server Technologies Documentation
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This book is a database high availability reference. It describes Oracle database architectures and features as well as recommended practices that can help your business achieve high availability. It provides guidelines for choosing the appropriate high availability solution.

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

Audience

This book is intended for chief technology officers, information technology architects, database administrators, system administrators, network administrators, and application administrators who perform the following tasks:

- Plan data centers
- Implement data center policies
- Maintain high availability systems
- Plan and build high availability solutions

Organization

This document contains:

Part I, "Getting Started"

This part provides an overview of high availability (HA) and describes the Oracle features that can be used to achieve high availability.

Chapter 1, "Overview of High Availability"

This chapter defines high availability and the need for HA architecture and practices. It describes in general terms what is necessary to achieve high availability. It gives examples of outages and their impact on businesses. It also explains the scope of the book and how to use the book.

Chapter 2, "Determining Your High Availability Requirements"

This chapter describes service level agreements and business requirements. It provides guidelines for determining whether data loss is acceptable and discusses the performance and manageability impact of HA practices.

Part II, "Oracle Database High Availability Features, Architectures, and Policies"

This part explains what business requirements influence the decision to implement a high availability solution. After the essential factors have been identified, defined, and described, the factors are used to provide guidance about choosing a high availability architecture.

Chapter 3, "Oracle Database High Availability Features"

This chapter provides high-level descriptions of Oracle HA features.

Chapter 4, "High Availability Architectures"

This chapter describes validated HA architectures.

Chapter 5, "Operational Policies for High Availability"

This chapter describes operational best practices for HA.

Part III, "Configuring a Highly Available Oracle Environment"

This part describes how to configure the high availability architectures.

Chapter 6, "System and Network Configuration"

This chapter provides recommendations for configuring the subcomponents that make up the database server tier and the network.

Chapter 7, "Oracle Configuration Best Practices"

This chapter recommends Oracle configuration and best practices for the database, Oracle Real Application Clusters, Oracle Data Guard, Maximum Availability Architecture, backup and recovery, and fast application failover.

Part IV, "Managing a Highly Available Oracle Environment"

This part describes how to manage an HA Oracle environment.

Chapter 8, "Using Oracle Enterprise Manager for Monitoring and Detection"

This chapter describes how to monitor and detect system availability. It emphasizes Oracle Enterprise Manager.

Chapter 9, "Recovering from Outages"

This chapter contains a decision matrix for determining what actions to take for specific outages.

Chapter 10, "Detailed Recovery Steps"

This chapter contains detailed steps for recovering from the outages described in Chapter 9, "Recovering from Outages".

Chapter 11, "Restoring Fault Tolerance"

This chapter describes the following types of repair: restoring failed nodes in a Real Application Cluster, restoring the standby database after a failover, restoring fault tolerance after secondary site or clusterwide scheduled outage, restoring fault tolerance after a standby database data failure, restoring fault tolerance after the production database is activated, and restoring fault rolerance after dual failures.

Appendix A, "Hardware Assisted Resilient Data (HARD) Initiative"

This appendix contains information about the Hardware Assisted Resilient Data (HARD) initiative.

Appendix B, "Database SPFILE and Oracle Net Configuration File Samples"

This appendix contains database `SPFILE` and Oracle Net configuration file samples.

Related Documentation

For more information, see the Oracle database documentation set. These books may be of particular interest:

- *Oracle Data Guard Concepts and Administration*
- *Oracle Real Application Clusters Deployment and Performance Guide*
- *Oracle Database Backup and Recovery Advanced User's Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Application Server 10g High Availability Guide*

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

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

`http://otn.oracle.com/membership/index.html`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/documentation/index.html`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	DECIMAL (*digits* [, *precision*])
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	{ENABLE | DISABLE}
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	CREATE TABLE ... AS *subquery*; SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

Convention	Meaning	Example

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Part I

Getting Started

This part provides an overview of high availability and determining your high availability requirements.

This part contains the following chapters:

- Chapter 1, "Overview of High Availability"
- Chapter 2, "Determining Your High Availability Requirements"

1

Overview of High Availability

This chapter contains the following sections:

- Introduction to High Availability
- What is Availability?
- Importance of Availability
- Causes of Downtime
- What Does This Book Contain?
- Who Should Read This Book?

Introduction to High Availability

Databases and the Internet have enabled worldwide collaboration and information sharing by extending the reach of database applications throughout organizations and communities. This reach emphasizes the importance of high availability (HA) in data management solutions. Both small businesses and global enterprises have users all over the world who require access to data 24 hours a day. Without this data access, operations can stop, and revenue is lost. Users, who have become more dependent upon their solutions, now demand service-level agreements from their Information Technology (IT) departments and solutions providers. Increasingly, availability is measured in dollars, euros, and yen, not just in time and convenience.

Enterprises have used their IT infrastructure to provide a competitive advantage, increase productivity, and empower users to make faster and more informed decisions. However, with these benefits has come an increasing dependence on that infrastructure. If a critical application becomes unavailable, then the entire business can be in jeopardy. Revenue and customers can be lost, penalties can be owed, and bad publicity can have a lasting effect on customers and a company's stock price. It is critical to examine the factors that determine how your data is protected and maximize the availability to your users.

What is Availability?

Availability is the degree to which an application or service is available when, and with the functionality, users expect. Availability is measured by the perception of an application's end user. End users experience frustration when their data is unavailable, and they do not understand or care to differentiate between the complex components of an overall solution. Performance failures due to higher than expected usage create the same havoc as the failure of critical components in the solution.

Reliability, recoverability, and continuous operations are three characteristics of a highly available solution:

- **Reliability**: Reliable hardware is one component of an HA solution. Reliable software, including the database, web servers, and application, is as critical to implementing a highly available solution.
- **Recoverability**: There may be many choices in recovering from a failure if one occurs. It is important to determine what types of failures may occur in your high availability environment and how to recover from those failure in the time that meets your business requirements. For example, if a critical table is accidentally deleted from the database, what action would you take to recover

it? Does your architecture provide you the ability to recover in the time specified in a service level agreement (SLA)?

- **Error Detection**: If a component in your architecture fails, then fast detection is another essential component in recovering from a possible unexpected failure. While you may be able to recover quickly from an outage, if it takes an additional 90 minutes to discover the problem, then you may not meet your SLA. Monitoring the health of your environment requires reliable software to view it quickly and the ability to notify the DBA of a problem.
- **Continuous operations**: Continuous access to your data is essential when very little or no downtime is acceptable to perform maintenance activities. Such activities as moving a table to another location within the database or even adding additional CPUs to your hardware should be transparent to the end user in an HA architecture.

An HA architecture should have the following characteristics:

- Be transparent to most failures
- Provide built-in preventative measures
- Provide proactive monitoring and fast detection of failures
- Provide fast recoverability
- Automate the recovery operation
- Protect the data so that there is minimal or no data loss
- Implement the operational best practices to manage your environment
- Provide the HA solution to meet your SLA

Importance of Availability

The importance of high availability varies among applications. However, the need to deliver increasing levels of availability continues to accelerate as enterprises re-engineer their solutions to gain competitive advantage. Most often these new solutions rely on immediate access to critical business data. When data is not available, the operation can cease to function. Downtime can lead to lost productivity, lost revenue, damaged customer relationships, bad publicity, and lawsuits.

If a mission-critical application becomes unavailable, then the enterprise is placed in jeopardy. It is not always easy to place a direct cost on downtime. Angry customers, idle employees, and bad publicity are all costly, but not directly measured in

currency. On the other hand, lost revenue and legal penalties incurred because SLA objectives are not met can easily be quantified. The cost of downtime can quickly grow in industries that are dependent upon their solutions to provide service.

Other factors to consider in the cost of downtime are the maximum tolerable length of a single unplanned outage, and the maximum frequency of allowable incidents. If the event lasts less than 30 seconds, then it may cause very little impact and may be barely perceptible to end users. As the length of the outage grows, the effect grows from annoyance at a minor problem into a big problem, and possibly into legal action. When designing a solution, it is important to take into account these issues and to determine the true cost of downtime. An organization should then spend reasonable amounts of money to build solutions that are highly available, yet whose cost is justified. High availability solutions are effective insurance policies.

Oracle makes high availability solutions available to every customer regardless of size. Small workgroups and global enterprises alike are able to extend the reach of their critical business applications. With Oracle and the Internet, applications and their data are now reliably accessible everywhere, at any time.

Causes of Downtime

One of the challenges in designing an HA solution is examining and addressing all the possible causes of downtime. It is important to consider causes of both unplanned and planned downtime. Unplanned downtime includes computer failures and data failures. Data failures can be caused by storage failure, human error, corruption, and site failure. Planned downtime includes system changes and data changes. Planned downtime can be just as disruptive to operations, especially in global enterprises that support users in multiple time zones.

> **See Also:** Chapter 9, "Recovering from Outages" for a more detailed discussion of unplanned and planned outages

What Does This Book Contain?

Choosing and implementing the architecture that best fits your availability requirements can be a daunting task. This architecture must encompass redundancy across all components, achieve fast client failover for all types of downtime, provide consistent high performance, and provide protection from user errors, corruptions, and site disasters, while being easy to deploy, manage, and scale.

This book describes several HA architectures and provides guidelines for choosing the one that is best for your requirements. It also describes practices that are

essential to maintaining an HA environment regardless of the architecture that is chosen. It also describes types of outages and provides a blueprint for detecting and resolving the outages.

Who Should Read This Book?

Knowledge of the Oracle Database server, Real Application Clusters and Oracle Data Guard terminology is required to understand the configuration and implementation details.

See Also:

- *Oracle Database Concepts*
- The Oracle Data Guard documentation set
- The Real Application Clusters documentation set

Chief technology officers and information technology architects can benefit from reading the following chapters:

- Chapter 2, "Determining Your High Availability Requirements"
- Chapter 4, "High Availability Architectures"

Information technology architects can also find essential information in the following chapters:

- Chapter 5, "Operational Policies for High Availability"
- Chapter 3, "Oracle Database High Availability Features"

Database administrators can find useful information in the following chapters:

- Chapter 4, "High Availability Architectures"
- Chapter 5, "Operational Policies for High Availability"
- Chapter 6, "System and Network Configuration"
- Chapter 7, "Oracle Configuration Best Practices"
- Chapter 8, "Using Oracle Enterprise Manager for Monitoring and Detection"
- Chapter 9, "Recovering from Outages"
- Chapter 11, "Restoring Fault Tolerance"

Network administrators and application administrators can find useful information in the following chapters:

- Chapter 4, "High Availability Architectures"
- Chapter 5, "Operational Policies for High Availability"
- Chapter 6, "System and Network Configuration"
- Chapter 7, "Oracle Configuration Best Practices"
- Chapter 9, "Recovering from Outages"

2

Determining Your High Availability Requirements

This chapter includes the following topics:

- Why It Is Important to Determine High Availability Requirements
- Analysis Framework for Determining High Availability Requirements
- Choosing a High Availability Architecture

Why It Is Important to Determine High Availability Requirements

The interconnected nature of today's global businesses demands continuous availability for more of the business components. However, a business that is designing and implementing an HA strategy must perform a thorough analysis and have a complete understanding of the business drivers that require high availability, because implementing high availability is costly. It may involve critical tasks such as:

- Retiring legacy systems
- Investment in more sophisticated and robust systems and facilities
- Redesign of the overall IT architecture to adapt to this high availability model
- Redesign of business processes
- Hiring and training of personnel

Higher degrees of availability reduce downtime significantly, ash shown in the following table:

Availability Percentage	Approximate Downtime Per Year
95%	18 days
99%	4 days
99.9%	9 hours
99.99%	1 hour
99.999%	5 minutes

Businesses with higher availability requirements must deploy more fault-tolerant, redundant systems for their business components and have a larger investment in IT staff, processes, and services to ensure that the risk of business downtime is minimized.

An analysis of the business requirements for high availability and an understanding of the accompanying costs enables an optimal solution that meets the needs of the business managers to be available as much as possible within financial and resource limitations of the business. This chapter provides a simple framework that can be used effectively to evaluate the high availability requirements of a business.

Analysis Framework for Determining High Availability Requirements

The elements of this analysis framework are:

- Business Impact Analysis
- Cost of Downtime
- Recovery Time Objective
- Recovery Point Objective

Business Impact Analysis

A rigorous business impact analysis identifies the critical business processes within an organization, calculates the quantifiable loss risk for unplanned and planned IT outages affecting each of these business processes, and outlines the less tangible impacts of these outages. It takes into consideration essential business functions, people and system resources, government regulations, and internal and external business dependencies. This analysis is done using objective and subjective data gathered from interviews with knowledgeable and experienced personnel, reviewing business practice histories, financial reports, IT systems logs, and so on.

The business impact analysis categorizes the business processes based on the severity of the impact of IT-related outages. For example, consider a semiconductor manufacturer, with chip design centers located worldwide. An internal corporate system providing access to human resources, business expenses and internal procurement is not likely to be considered as mission-critical as the internal e-mail system. Any downtime of the e-mail system is likely to severely affect the collaboration and communication capabilities among the global R&D centers, causing unexpected delays in chip manufacturing, which in turn will have a material financial impact on the company.

In a similar fashion, an internal knowledge management system is likely to be considered mission-critical for a management consulting firm because the business of a client-focused company is based on internal research accessibility for its consultants and knowledge workers. The cost of downtime of such a system is extremely high for this business. This leads us to the next element in the high availability requirements framework: cost of downtime.

Cost of Downtime

A well-implemented business impact analysis provides insights into the costs that result from unplanned and planned downtimes of the IT systems supporting the

various business processes. Understanding this cost is essential because this has a direct influence on the high availability technology chosen to minimize the downtime risk.

Various reports have been published, documenting the costs of downtime across industry verticals. These costs range from millions of dollars per hour for brokerage operations and credit card sales, to tens of thousands of dollars per hour for package shipping services.

While these numbers are staggering, the reasons are quite obvious. The Internet has brought millions of customers directly to the businesses' electronic storefronts. Critical and interdependent business issues such as customer relationships, competitive advantages, legal obligations, industry reputation, and shareholder confidence are even more critical now because of their increased vulnerability to business disruptions.

Recovery Time Objective

A business impact analysis, as well as the calculated cost of downtime, provides insights into the recovery time objective (RTO), an important statistic in business continuity planning. It is defined as the maximum amount of time that an IT-based business process can be down before the organization starts suffering significant material losses. RTO indicates the downtime tolerance of a business process or an organization in general.

The RTO requirements are proportional to the mission-critical nature of the business. Thus, for a system running a stock exchange, the RTO is zero or very near to zero.

An organization is likely to have varying RTO requirements across its various business processes. Thus, for a high volume e-commerce Web site, for which there is an expectation of rapid response times and for which customer switching costs are very low, the web-based customer interaction system that drives e-commerce sales is likely to have an RTO close to zero. However, the RTO of the systems that support the backend operations such as shipping and billing can be higher. If these backend systems are down, then the business may resort to manual operations temporarily without a significantly visible impact.

A systems statistic related to RTO is the network recovery objective (NRO), which indicates the maximum time that network operations can be down for a business. Components of network operations include communication links, routers, name servers, load balancers, and traffic managers. NRO impacts the RTO of the whole organization because individual servers are useless if they cannot be accessed when the network is down.

Recovery Point Objective

Recovery point objective (RPO) is another important statistic for business continuity planning and is calculated through an effective business impact analysis. It is defined as the maximum amount of data an IT-based business process may lose before causing detrimental harm to the organization. RPO indicates the data-loss tolerance of a business process or an organization in general. This data loss is often measured in terms of time, for example, 5 hours or 2 days worth of data loss.

A stock exchange where millions of dollars worth of transactions occur every minute cannot afford to lose any data. Thus, its RPO must be zero. Referring to the e-commerce example, the web-based sales system does not strictly require an RPO of zero, although a low RPO is essential for customer satisfaction. However, its backend merchandising and inventory update system may have a higher RPO; lost data in this case can be re-entered.

Choosing a High Availability Architecture

Using the high availability analysis framework, a business can complete a business impact analysis, identify and categorize the critical business processes that have the high availability requirements, formulate the cost of downtime, and establish RTO and RPO goals for these various business processes.

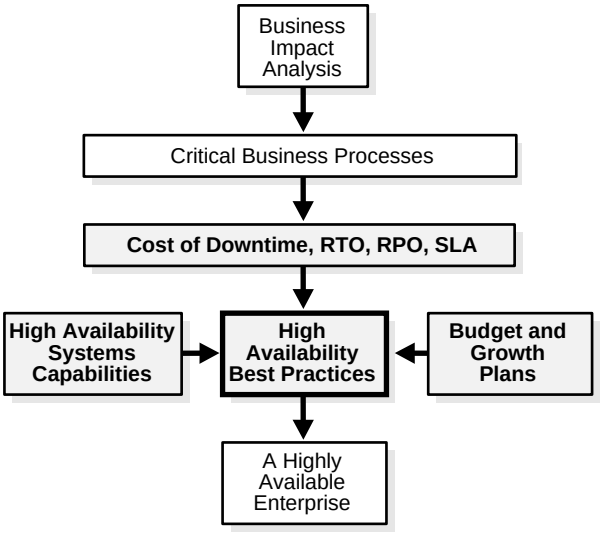
This enables the business to define service level agreements (SLAs) in terms of high availability for critical aspects of its business. For example, it can categorize its businesses into several HA tiers:

- Tier 1 business processes have maximum business impact. They have the most stringent HA requirements, with RTO and RPO close to zero, and the systems supporting it need to be available on a continuous basis. For a business with a high-volume e-commerce presence, this may be the web-based customer interaction system.
- Tier 2 business processes can have slightly relaxed HA and RTO/RPO requirements. The second tier of an e-commerce business may be their supply chain / merchandising systems. For example, these systems do not need to maintain 99.999% availability and may have nonzero RTO/RPO values. Thus, the HA systems and technologies chosen to support these two tiers of businesses are likely to be different.
- Tier 3 business processes may be related to internal development and quality assurance processes. Systems support these processes need not have the rigorous HA requirements of the other tiers.

The next step for the business is to evaluate the capabilities of the various HA systems and technologies and choose the ones that meet its SLA requirements, within the guidelines as dictated by business performance issues, budgetary constraints, and anticipated business growth.

Figure 2–1 illustrates this process.

Figure 2–1 Planning and Implementing a Highly Available Enterprise



The following sections provide further details about this methodology:

- HA Systems Capabilities
- Business Performance, Budget and Growth Plans
- High Availability Best Practices

See Also:

- Table 4–3, "Service Level Agreements and Recommended HA Architectures" on page 4-14
- "Choosing the Correct HA Architecture" on page 4-12

HA Systems Capabilities

A broad range of high availability and business continuity solutions exists today. As the sophistication and scope of these systems increase, they make more of the IT infrastructure, such as the data storage, server, network, applications, and facilities, highly available. They also reduce RTO and RPO from days to hours, or even to minutes. But increased availability comes with an increased cost, and on some occasions, with an increased impact on systems performance.

Organizations need to carefully analyze the capabilities of these HA systems and map their capabilities to the business requirements to make sure they have an optimal combination of HA solutions to keep their business running. Consider the business with a significant e-commerce presence as an example.

For this business, the IT infrastructure that supports the system that customers encounter, the core e-commerce engine, needs to be highly available and disaster-proof. The business may consider clustering for the web servers, application servers and the database servers serving this e-commerce engine. With built-in redundancy, clustered solutions eliminate single points of failure. Also, modern clustering solutions are application-transparent, provide scalability to accommodate future business growth, and provide load-balancing to handle heavy traffic. Thus, such clustering solutions are ideally suited for mission-critical high-transaction applications.

The data that supports the high volume e-commerce transactions must be protected adequately and be available with minimal downtime if unplanned and planned outages occur. This data should not only be backed up at regular intervals at the local data centers, but should also be replicated to databases at a remote data center connected over a high-speed, redundant network. This remote data center should be equipped with secondary servers and databases readily available, and be synchronized with the primary servers and databases. This gives the business the capability to switch to these servers at a moment's notice with minimal downtime if there is an outage, instead of waiting for hours and days to rebuild servers and recover data from backed-up tapes.

Maintaining synchronized remote data centers is an example where redundancy is built along the entire system’s infrastructure. This may be expensive. However, the mission-critical nature of the systems and the data it protects may warrant this expense. Considering another aspect of the business: for example, the high availability requirements are less stringent for systems that gather clickstream data and perform data mining. The cost of downtime is low, and the RTO and RPO requirements for this system could be a few days, because even if this system is down and some data is lost, that will not have a detrimental effect on the business.

While the business may need powerful machines to perform data mining, it does not need to mirror this data on a real-time basis. Data protection may be obtained by simply performing regularly scheduled backups, and archiving the tapes for offsite storage.

For this e-commerce business, the back-end merchandising and inventory systems are expected to have higher HA requirements than the data mining systems, and thus they may employ technologies such as local mirroring or local snapshots, in addition to scheduled backups and offsite archiving.

The business should employ a management infrastructure that performs overall systems management, administration and monitoring, and provides an executive dashboard. This management infrastructure should be highly available and fault-tolerant.

Finally, the overall IT infrastructure for this e-commerce business should be extremely secure, to protect against malicious external and internal electronic attacks.

Business Performance, Budget and Growth Plans

High availability solutions must also be chosen keeping in mind business performance issues. For example, a business may use a zero-data-loss solution that synchronously mirrors every transaction on the primary database to a remote database. However, considering the speed-of-light limitations and the physical limitations associated with a network, there will be round-trip-delays in the network transmission. This delay increases with distance, and varies based on network bandwidth, traffic congestion, router latencies, and so on. Thus, this synchronous mirroring, if performed over large WAN distances, may impact the primary site performance. Online buyers may notice these system latencies and be frustrated with long system response times; they may go somewhere else for their purchases. This is an example where the business must make a trade-off between having a zero data loss solution and maximizing system performance.

High availability solutions must also be chosen keeping in mind financial considerations and future growth estimates. It is tempting to build redundancies throughout the IT infrastructure and claim that the infrastructure is completely failure-proof. However, going overboard with such solutions may not only lead to budget overruns, it may lead to an unmanageable and unscalable combination of solutions that are extremely complex and expensive to integrate and maintain.

An HA solution that has very impressive performance benchmark results may look good on paper. However, if an investment is made in such a solution without a careful analysis of how the technology capabilities match the business drivers, then

a business may end up with a solution that does not integrate well with the rest of the system infrastructure, has annual integration and maintenance costs that easily exceed the upfront license costs, and forces a vendor lock-in. Cost-conscious and business-savvy CIOs must invest only in solutions that are well-integrated, standards-based, easy to implement, maintain and manage, and have a scalable architecture for accommodating future business growth.

High Availability Best Practices

Choosing and implementing the architecture that best fits the availability requirements of a business can be a daunting task. This architecture must encompass appropriate redundancy, provide adequate protection from all types of outages, ensure consistent high performance and robust security, while being easy to deploy, manage, and scale. Needless to mention, this architecture should be driven by well-understood business requirements.

To build, implement and maintain such an architecture, a business needs high availability best practices that involve both technical and operational aspects of its IT systems and business processes. Such a set of best practices removes the complexity of designing an HA architecture, maximizes availability while using minimum system resources, reduces the implementation and maintenance costs of the HA systems in place, and makes it easy to duplicate the high availability architecture in other areas of the business.

An enterprise with a well-articulated set of high availability best practices that encompass HA analysis frameworks, business drivers and system capabilities, will enjoy an improved operational resilience and enhanced business agility. The remaining chapters in this book will provide technical details on the various high availability technologies offered by Oracle, along with best practice recommendations on configuring and using such technologies.

See Also: Chapter 5, "Operational Policies for High Availability"

Part II

Oracle Database High Availability Features, Architectures, and Policies

This part highlights Oracle Database high availability features and describes architectures that use Oracle high availability features and products. It also describes operation policies that are an essential part of maintaining high availability.

This part contains the following chapters:

- Chapter 3, "Oracle Database High Availability Features"
- Chapter 4, "High Availability Architectures"
- Chapter 5, "Operational Policies for High Availability"

3

Oracle Database High Availability Features

This chapter describes Oracle Database high availability features. It includes the following topics:

- Oracle Real Application Clusters
- Oracle Data Guard
- Oracle Streams
- Online Reorganization
- Transportable Tablespaces
- Automatic Storage Management
- Flashback Technology
- Dynamic Reconfiguration
- Oracle Fail Safe
- Recovery Manager
- Flash Recovery Area
- Hardware Assisted Resilient Data (HARD) Initiative

Oracle Real Application Clusters

Oracle Real Application Clusters (RAC) enables multiple instances that are linked by an interconnect to share access to an Oracle database. This enables RAC to provide high availability, scalability, and redundancy during failures. RAC provides scalability without requiring application code changes.

RAC accommodates all system types, from read-only data warehouse (DSS) systems to update-intensive online transaction processing (OLTP) systems as well as systems that combine both DSS and OLTP. Typical RAC environments are configured with symmetric multi-processors.

RAC provides the following benefits:

- Node and instance failover in seconds
- Integrated and intelligent connection and service failover across various instances
- Planned node, instance, and service switchover and switchback
- Rolling patch upgrades
- Multiple active instance availability and scalability across multiple nodes
- Comprehensive manageability integrating database and cluster features

See Also: *Oracle Real Application Clusters Quick Start*

Oracle Data Guard

Oracle Data Guard provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases to enable production Oracle databases to survive failures, disasters, errors, and data corruptions. Data Guard maintains these standby databases as transactionally consistent copies of the production database. Then, if the production database becomes unavailable because of a planned or an unplanned outage, Data Guard can switch any standby database to the production role, thus minimizing the downtime associated with the outage. Data Guard can be used with traditional backup, restoration, and cluster technology to provide a high level of data protection and data availability.

A Data Guard configuration consists of one production database and one or more physical or logical standby databases. The databases in a Data Guard configuration are connected by Oracle Net and may be dispersed geographically. There are no restrictions on where the databases are located if they can communicate with each other. For example, you can have a standby database in the same building as your

primary database to help manage planned downtime and two or more standby databases in other locations for use in disaster recovery.

Data Guard provides the following benefits:

- Disaster recovery, data protection and high availability

  Data Guard provides an efficient and comprehensive disaster recovery and high availability solution. Easy-to-manage switchover and failover capabilities allow role reversals between primary and standby databases, minimizing the downtime of the primary database for planned and unplanned outages.

- Complete data protection

  With standby databases, Data Guard guarantees no data loss, even in the face of unforeseen disasters. A standby database provides a safeguard against data corruption and user errors. Storage level physical corruptions on the primary database do not propagate to the standby database. Similarly, logical corruptions or user errors that cause the primary database to be permanently damaged can be resolved. Finally, the redo data is validated when it is applied to the standby database.

- Efficient use of system resources

  The standby database tables that are updated with redo data received from the primary database can be used for other tasks such as backups, reporting, summations, and queries, thereby reducing the primary database workload necessary to perform these tasks, saving valuable CPU and I/O cycles. With a logical standby database, users can perform normal data manipulation on tables in schemas that are not updated from the primary database. A logical standby database can remain open while the tables are updated from the primary database, and the tables are simultaneously available for read-only access. Finally, additional indexes and materialized views can be created on the maintained tables for better query performance and to suit specific business requirements.

- Flexibility in data protection to balance availability against performance requirements

  Data Guard offers maximum protection, maximum availability, and maximum performance modes to help enterprises balance data availability against system performance requirements.

- Automatic gap detection and resolution

  If connectivity is lost between the primary and one or more standby databases (for example, due to network problems), redo data being generated on the

primary database cannot be sent to those standby databases. After connectivity is reestablished, the missing log files (referred to as a gap) are automatically detected by Data Guard, which then automatically transmits the missing log files to the standby databases. The standby databases are resynchronized with the primary database, without manual intervention by the DBA.

- Centralized and simple management

  The Data Guard broker provides a graphical user interface and a command-line interface to automate management and operational tasks across multiple databases in a Data Guard configuration. The broker also monitors all of the systems within a single Data Guard configuration.

- Integration with the Oracle database

See Also: *Oracle Data Guard Concepts and Administration*

Oracle Streams

Oracle Streams enables the propagation and management of data, transactions, and events in a data stream, either within a database or from one database to another. Streams provides a set of elements that allow users to control what information is put into a data stream, how the stream is routed from node to node, what happens to events in the stream as they flow into each node, and how the stream terminates.

Streams can be used to replicate a database or a subset of a database. This enables users and applications to simultaneously update data at multiple locations. If a failure occurs at one of the locations, then users and applications at the surviving sites can continue to access and update data.

Streams can be used to build distributed applications that replicate changes at the application level using message queuing. If an application fails, then the surviving applications can continue to operate and provide access to data through locally maintained copies.

Streams provides granularity and control over what is replicated and how it is replicated. It supports bidirectional replication, data transformations, subsetting, custom apply functions, and heterogeneous platforms. It also gives users complete control over the routing of change records from the primary database to a replica database.

See Also: *Oracle Streams Concepts and Administration*

Online Reorganization

Database administrators can perform a variety of online operations to table definitions, including online reorganization of heap-organized tables. This makes it possible to reorganize a table while users have full access to it.

This online architecture provides the following benefits:

- Any physical attribute of the table can be changed online. The table can be moved to a new location. The table can be partitioned. The table can be converted from one type of organization (such as heap-organized) to another (such as index-organized).
- Many logical attributes can also be changed. Column names, types, and sizes can be changed. Columns can be added, deleted, or merged. One restriction is that the primary key of the table cannot be modified.
- Secondary indexes on index-organized tables can be created and rebuilt online. Secondary indexes support efficient use of block hints (physical guesses). Invalid physical guesses can be repaired online.
- Indexes can be created online and analyzed at the same time. Online repair of the physical guess component of logical rowids (used in secondary indexes and in the mapping table for index-organized tables) also can be used.
- The physical guess component of logical rowids stored in secondary indexes on index-organized tables can be repaired. This enables online repair of invalid physical guesses.
- You can reorganize an index-organized table or an index-organized table partition without rebuilding its secondary indexes. This results in a short reorganization maintenance window.
- You can reorganize an index-organized table online. Along with online reorganization of secondary indexes, this capability eliminates the reorganization maintenance window.

See Also: *Oracle Database Administrator's Guide*

Transportable Tablespaces

The transportable tablespace feature enables users to quickly move a tablespace across Oracle databases. It is the most efficient way to move bulk data between databases.

Moving data using transportable tablespaces can be much faster than performing either an export/import or unload/load of the same data. This is because transporting a tablespace requires only the copying of datafiles and integrating the tablespace structural information. You can also use transportable tablespaces to move index data, thereby avoiding the index rebuilds you would have to perform when importing or loading table data.

You can transport tablespaces across platforms. This functionality can be used to:

- Provide an easier and more efficient means for content providers to publish structured data and distribute it to customers running Oracle on a different platform
- Simplify the distribution of data from a data warehouse environment to data marts which are often running on smaller systems with a different platform
- Enable the sharing of read-only tablespaces across a heterogeneous cluster
- Allow a database to be migrated from one platform to another

Most platforms are supported for cross-platform tablespace transport. You can query the V$TRANSPORTABLE_PLATFORM view to see the platforms that are supported and to determine their platform IDs and their endian format (byte ordering).

See Also: *Oracle Database Administrator's Guide*

Automatic Storage Management

Automatic storage management (ASM) automates and simplifies the optimal layout of datafiles, control files, redo log files, and flash recovery area files for both single instance and RAC databases. ASM is designed to work with any type of storage, from unmanaged disks to a SAN-based, intelligent storage array.

ASM maximizes performance by automatically distributing database files across all available disks. Database storage is automatically rebalanced whenever the storage configuration changes while the database remains online. You never need to manually relocate data to reclaim space because this approach eliminates storage fragmentation.

ASM provides data protection by maintaining redundant copies, or mirrors, of data.The protection and striping policy can be defined for each file to allow varying degrees of protection striping within the same set of disks.

ASM disk groups, which are comprised of disks and the files that reside on them, simplify storage administration by allowing a collection of disks to be managed as a

single unit. ASM failure groups allow the disks in a disk group to be subdivided into sets of disks that share a common resource whose failure needs to be tolerated. An example of a failure group is a string of SCSI disks connected to a common SCSI controller.

See Also: *Oracle Database Administrator's Guide*

Flashback Technology

Human errors are difficult to avoid and can be particularly difficult to recover from without aplanning and the right technology. Such errors can result in logical data corruption or cause downtime of one or more components of the IT infrastructure. While it is relatively simple to rectify the failure of an individual component, detection and repair of logical data corruption, such as accidental deletion of valuable data, is a time-consuming operation that causes enormous loss of business productivity.

Flashback technology provides a set of features to view and rewind data back and forth in time. The flashback features offer the capability to query past versions of schema objects, query historical data, perform change analysis. and perform self-service repair to recover from logical corruptions while the database is online.

Flashback technology provides a SQL interface to quickly analyze and repair human errors. Flashback provides fine-grained analysis and repair for localized damage such as deleting the wrong customer order. Flashback technology also enables correction of more widespread damage, yet does it quickly to avoid long downtime. Flashback technology is unique to the Oracle Database and supports recovery at all levels including the row, transaction, table, tablespace, and database.

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide*
- *Oracle Database SQL Reference*

Oracle Flashback Query

Oracle Flashback Query enables you to specify a target time and then run queries against the database, viewing results as they would have appeared at that time. To recover from an unwanted change like an erroneous update to a table, you can choose a target time before the error and run a query to retrieve the contents of the lost rows.

Oracle Flashback Version Query

Oracle Flashback Version Query retrieves metadata and historical data for a specific time interval. You can view all the rows of a table that ever existed during a specific time interval. Metadata about the different versions of rows includes start and end time, type of change operation, and identity of the transaction that created the row version.

Oracle Flashback Transaction Query

Oracle Flashback Transaction Query retrieves metadata and historical data for a specific transaction or for all transactions within a specific time interval. You can also obtain the SQL code to undo the changes to particular rows affected by a transaction. You typically use Flashback Transaction Query with Flashback Version Query, which provides the transaction IDs for the rows of interest.

Oracle Flashback Table

Oracle Flashback Table recovers a table to its state at a previous point in time. You can restore table data while the database is online, undoing changes to only the specified table.

Oracle Flashback Drop

Oracle Flashback Drop recovers a dropped table. This reverses the effects of a DROP TABLE statement.

Oracle Flashback Database

Oracle Flashback Database provides a more efficient alternative to database point-in-time recovery. When you use Flashback Database, your current datafiles revert to their contents at a past time. The result is much like the result of a point-in-time recovery using datafile backups and redo logs, but you do not have to restore datafiles from backup, and you do not have to re-apply as many individual changes in the redo logs as you would have to do in conventional media recovery.

Dynamic Reconfiguration

The Oracle database includes several features that enable changes to be made to the instance configuration dynamically. For example, the dynamic SGA infrastructure can be used to alter an instance's memory usage. It enables the size of the buffer

cache, the shared pool, the large pool, and the process-private memory to be changed without shutting down the instance. Oracle also provides transparent management of working memory for SQL execution by self-tuning the initialization runtime parameters that control allocation of private memory.

Another type of dynamic reconfiguration occurs when Oracle polls the operating system to detect changes in the number of available CPUs and reallocates internal resources.

In addition, some initialization parameters can be changed without shutting down the instance. You can use the `ALTER SESSION` statement to change the value of a parameter during a session. You can use the `ALTER SYSTEM` statement to change the value of a parameter in all sessions of an instance for the duration of the instance.

See Also:

- *Oracle Database Concepts* for more information about the dynamic SGA infrastructure
- *Oracle Database Reference* for information about dynamic initialization parameters

Oracle Fail Safe

Oracle Fail Safe is a software option that works with Microsoft Cluster Server (MSCS) to provide highly available business solutions on Microsoft clusters. A Microsoft cluster is a configuration of two or more Windows systems that appears to network users as a single, highly available system.

Oracle Fail Safe works with MSCS cluster software to provide high availability for applications and single-instance databases running on a cluster. When a node fails, the cluster software fails over to the surviving node based on parameters that you configure using Oracle Fail Safe.

With Oracle Fail Safe, you can reduce downtime for single-instance Oracle databases, Oracle HTTP servers, and almost any application that can be configured as a Windows service.

See Also: Oracle Fail Safe documentation at `http://otn.oracle.com/docs/tech/windows/failsafe/index.html`

Recovery Manager

Database backup, restoration, and recovery are essential processes underlying any high availability system. Imagine the potential for lost revenue, customer dissatisfaction, and unrecoverable information caused by a disk failure or human error. A well-designed and well-implemented backup and recovery strategy is a cornerstone for every database deployment, making it possible to restore and recover all or part of a database without data loss.

Recovery Manager (RMAN) is Oracle's utility to manage the backup and, more importantly, the recovery of the database. It eliminates operational complexity while providing superior performance and availability of the database.

Recovery Manager determines the most efficient method of executing the requested backup, restoration, or recovery operation and then executes these operations with the Oracle database server. Recovery Manager and the server automatically identify modifications to the structure of the database and dynamically adjust the required operation to adapt to the changes.

RMAN provides the following benefits:

- Block media recovery enables the datafile to remain online while fixing the block corruptions.
- Persistent RMAN configurations simplify backup and recovery operations.
- Retention policy ensures that relevant backups are retained.
- Resumable backup backs up files that were not backed up in a previous, failed backup operation
- Resumable restore decreases recovery time by restoring only files that require recovery.
- Backup optimization backs up only files that require a backup and have never been backed up.
- Restore optimization takes the guesswork out of restoring datafiles and archive logs. RMAN restores only if it is required.
- Automatic backup of the control file and the server parameter file means that backup metadata is available in times of database structural changes as well as media failure and disasters.
- Reporting features that answer the question, "Is my database recoverable?".
- Multiple block size backup support

- Online backup does not require the database to be placed into hot backup mode.
- You can perform fast incremental backups.
- You can reorganize and manage all recovery-related files such as backups, archived redo logs, and flashback logs.
- You can merge RMAN incremental backups and image copies in the backup to provide up to date recoverability.

See Also: *Oracle Database Backup and Recovery Basics*

Flash Recovery Area

The flash recovery area is a unified storage location for all recovery-related files and activities in an Oracle database. By defining one initialization parameter, all RMAN backups, archive logs, control file autobackups, and datafile copies are automatically written to a specified file system or automatic storage management disk group.

Making a backup to disk is faster because using the flash recovery area eliminates the bottleneck of writing to tape. More importantly, if database media recovery is required, then datafile backups are readily available. Restoration and recovery time is reduced because you do not need to find a tape and a free tape device to restore the needed datafiles and archive logs.

The flash recovery area provides:

- Unified storage location of related recovery files
- Management of the disk space allocated for recovery files
- Simplified database administration tasks
- Fast backup
- Fast restoration
- Reliability because the inherent reliability of the disk

Hardware Assisted Resilient Data (HARD) Initiative

Oracle has introduced the Hardware Assisted Resilient Data (HARD) Initiative, which is a program designed to prevent data corruptions before they happen. Data

corruptions are very rare, but when they happen, they can have a catastrophic effect on a database, and therefore a business.

Under the HARD Initiative, Oracle continues to work with selected system and storage vendors to build operating system and storage components that can detect corruptions early and prevent corrupted data from being written to disk. The kay approach is block checking where the storage subsystem validates the Oracle block contents. Implementation of this feature is transparent to the end user or DBA, regardless of the hardware vendor.

> **See Also:** Appendix A, "Hardware Assisted Resilient Data (HARD) Initiative"

4

High Availability Architectures

This chapter describes high availability architectures in an Oracle environment. It includes the following sections:

- Oracle Database High Availability Architectures
- Choosing the Correct HA Architecture
- Assessing Other Architectures

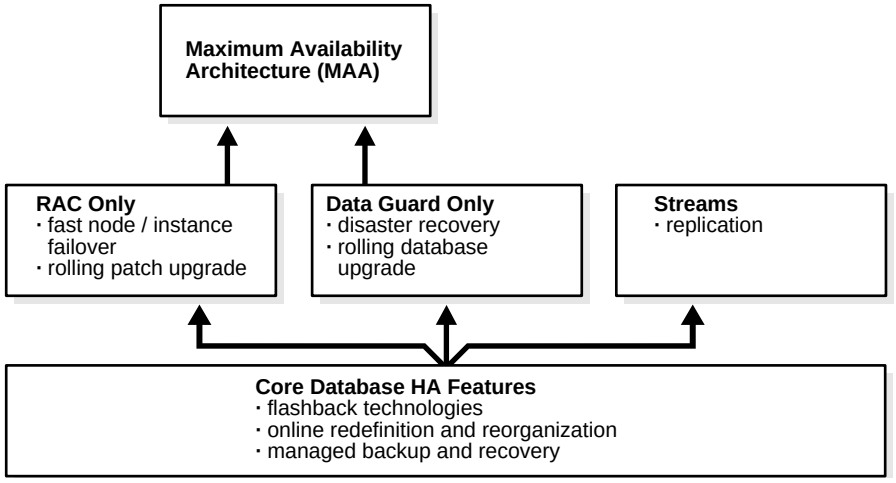
Oracle Database High Availability Architectures

This section defines the top five database architectures that address various high availability business needs. The architectures described are:

- **Database only**: This is a single instance production database on a standalone machine and uses all of the HA features that come with the Oracle database. Flashback technology can recover quickly from user and logical errors. Online redefinition and reorganization can reduce downtime for most scheduled maintenance activities. Oracle's sophisticated backup and recovery framework with RMAN can be customized for each customer's recovery requirements.
- **RAC only**: This involves a RAC database. RAC environments can provide continuous service for both planned and unplanned outages within the cluster. If a node or instance fails, then the application service fails over to existing RAC instances quickly and transparently.
- **Data Guard only**: The primary site contains a production database. The secondary site contains one or more standby databases. If there is a primary site failure or database failure or any failure that cannot be resolved quickly at the local site, then you can use Data Guard to fail over or switch over to a standby database.
- **Maximum Availability Architecture (MAA)**: The primary site contains a RAC production database. The secondary site contains a cluster that hosts both logical and physical standby databases. This architecture provides the most comprehensive set of solutions for both unscheduled and scheduled outages because it inherits the capabilities and advantages of both the RAC and the Data Guard architectures.
- **Streams**: The primary site contains a production database. The secondary site contains a replicated database using Streams technology. Both sites can be active, and transactions can be synchronized in both directions. The platforms can also be heterogeneous.

Oracle provides a wide array of HA architectural solutions. The "Database only" architecture is not highly available, but it contains many availability features and assets that are used by all other architectures. The "Database only" architecture is the starting point for most customers. "RAC only", "Data Guard only", and Streams architectures provide additional HA capabilities in addition to the "Database only" capabilities. MAA incorporates both RAC and Data Guard advantages and represents the architecture with maximum availability. Figure 4–1 illustrates the hierarchy of the different HA architectures.

Figure 4–1 Hierarchy of HA Architectures



The following sections describe each architecture and their advantages in more detail:

- "Database Only" Architecture
- "RAC Only" Architecture
- "Data Guard Only" Architecture
- Maximum Availability Architecture
- Streams Architecture

"Database Only" Architecture

Oracle provides HA features that can be used in any of the database architectures. These features make the standalone database on a single machine attractive and available. These features are described in Table 4–1. Chapter 9, "Recovering from Outages" describes when to use these features to resolve different outages, while Chapter 10, "Detailed Recovery Steps" contains details about how to use the features to recover.

Table 4–1 High Availability Features of the Oracle Database

Feature	Description
Reduced downtime for application and Oracle upgrades	■ One-step cloning of database objects for faster online redefinition during upgrades ■ Eliminates PL/SQL package invalidation after changes to underlying objects ■ Handles data definition language (DDL) locks on busy tables
Flashback capabilities to protect from user errors and logical corruptions	■ Flashback Version Query retrieves metadata and historical data for a specific time interval ■ Flashback Transaction Query retrieves metadata and historical data for a specific transaction or for all transactions within a specific time interval ■ Flashback Table recovers a table to its state at a previous point in time ■ Flashback Drop recovers a dropped table ■ Flashback Database provides a more efficient alternative to database point-in-time recovery
Online redefinition and reconfiguration for minimal or no scheduled downtime for changes to object structures or applications	■ Any physical attribute of a table can be changed online. The table can be moved to a new location. The table can be partitioned. The table can be converted from one type of organization (such as heap-organized) to another (such as index-organized). ■ Many logical attributes can be changed. Column names, types, and sizes can be changed. Columns can be added, deleted, or merged. (The primary key of the table cannot be modified.) ■ Secondary indexes can be created and rebuilt on index-organized tables (IOTs). Secondary indexes support efficient use of block hints (physical guesses). Invalid physical guesses can be repaired online. ■ Index can be created online and analyzed at the same time. ■ CLOB and BLOB datatypes have extended support.

Table 4–1 High Availability Features of the Oracle Database (Cont.)

Feature	Description
Manage backup and recovery operations automatically	■ Automated disk-based backup and recovery that provides a simplified and unified storage location for backups, archived redo log files, flashback logs, and other files required for recovery ■ RMAN backup capabilities such as backup standby control file support, simplified cataloging of backup files, simplified backups to disk, and improved incremental backups to avoid scanning an entire datafile ■ RMAN recovery capabilities such as automated file creation during recovery, simplified recovery through resetlogs, and the ability to flash back and recover to different points in time
Fast and efficient object re-creation features	Data Pump
Fast-start recovery during startup	Fast-start checkpointing can be used to reduce instance recovery time
Simple and integrated management framework with Oracle Enterprise Manager	■ Easy GUI for installation, set-up, and configuration ■ Integrated monitoring and management practices

"RAC Only" Architecture

"RAC only" architecture uses Real Application Clusters and is an inherently high availability system.The clusters that are typical of RAC environments can provide continuous service for both planned and unplanned outages. RAC build higher levels of availability on top of the standard Oracle features. All single instance HA features, such as flashback technologies and online reorganization, apply to RAC as well.

In addition to the standard Oracle features, RAC exploits the redundancy that is provided by clustering to deliver availability with *n* - 1 node failures in an *n*-node cluster. All users have access to all as long as there is one available node in the cluster.

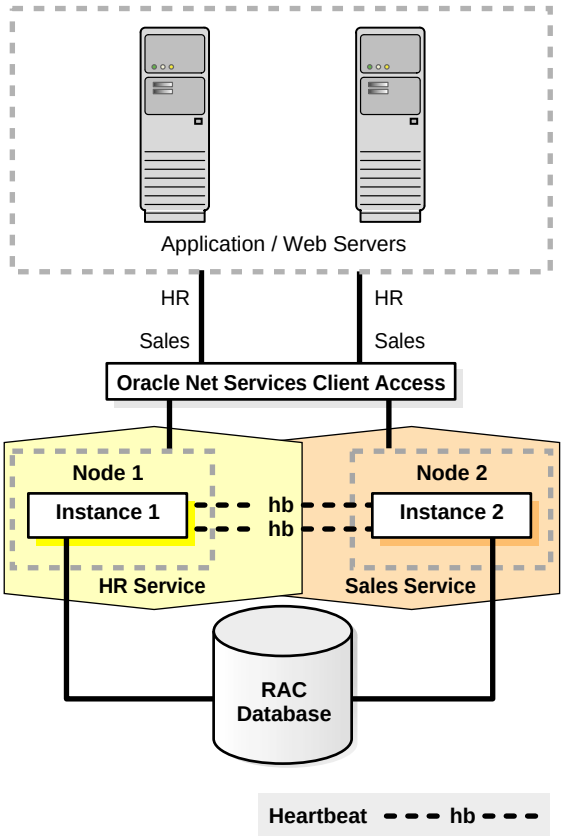
This architecture provides the following benefits:

- Fast node and instance failover (measured in seconds)
- Integrated and intelligent connection and service failover across various instances

- Planned node, instance, and service switchover and switchback
- Rolling patch upgrades
- Multiple active instance availability and scalability across multiple nodes
- Comprehensive manageability that integrates database and cluster features

Figure 4–2 shows "RAC only" architecture.

Figure 4–2 "RAC only" Architecture



"Data Guard Only" Architecture

Data Guard provides a rich set of HA solutions and addresses the requirements of the business community for a disaster recovery solution. Data Guard supports Data Guard Redo Apply and Data Guard SQL Apply technologies to enable two distinct kinds of standby databases: physical standby database and logical standby database.

Physical standby databases have provided HA solutions to thousands of applications since Oracle version 6. The introduction of Oracle Data Guard in Oracle8*i* and further enhancements in the Oracle9*i* and Oracle Database 10*g* releases have provided a simple and sophisticated HA suite. Data Guard physical standby architectures are the most efficient and simplest standby database environments. Logical standby databases, introduced in Oracle9*i*, enable recovery of transactions while permitting read and write operations to occur simultaneously in the same database. Logical standby databases require additional processing, but for some applications, they can be a very suitable HA solution or can complement other solutions. For some customers, both physical and logical standby databases can be used as part of an HA solution, which is recommended in MAA. The physical standby database becomes the initial switchover and failover target, while the logical standby database is used for reporting, queries, and additional processing to relieve the load on the production database. The logical standby database can also be targeted for switchovers, failovers, and rolling database upgrades.

Physical standby databases provide these advantages:

- Protection from user errors and logical corruptions
- Protection from disasters and site failures if located remotely
- Fast site and database failover (measured in minutes)
- Fast site and database planned switchovers for maintenance
- Backups can be taken from the physical standby database instead of the production database, relieving the load on the production database
- Read-only capability, resulting in better use of system resources

In addition to disaster recovery and data protection, logical standby databases provide the following benefits:

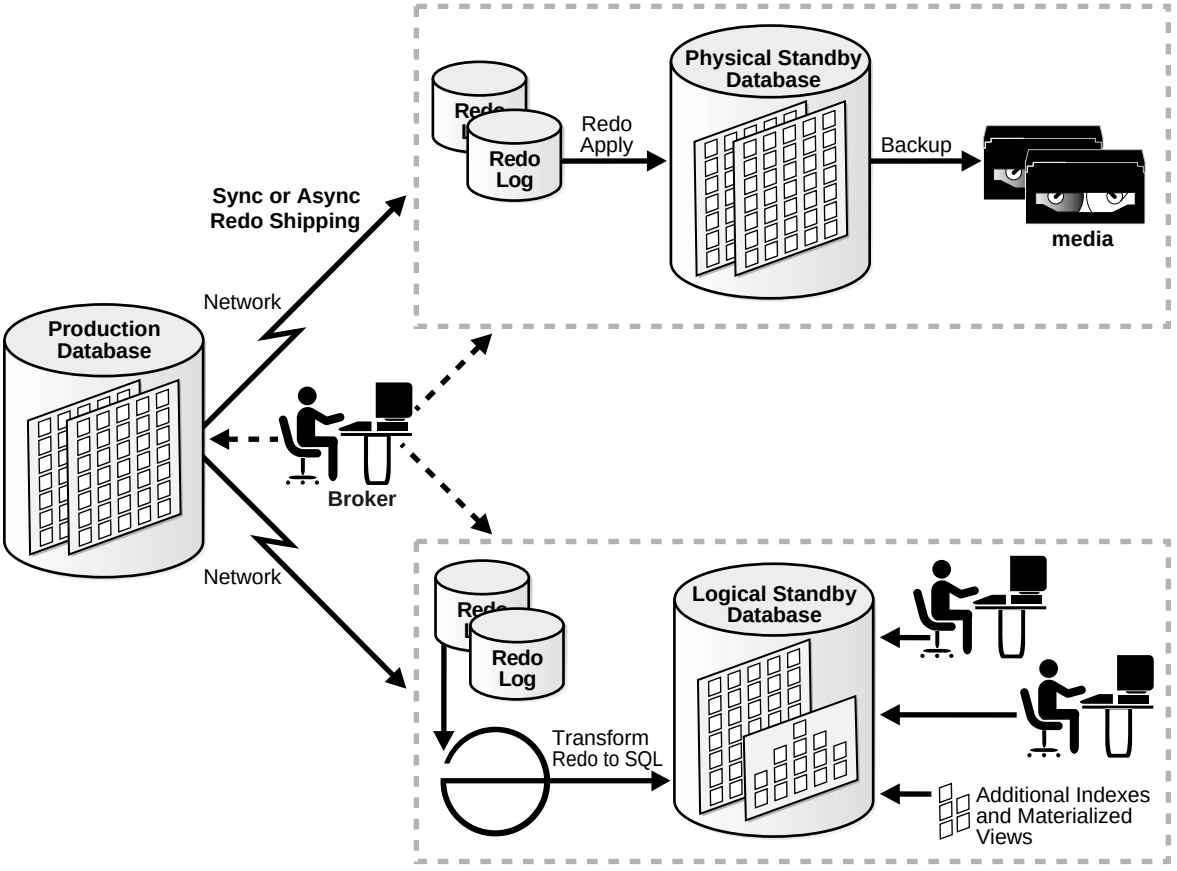
- Enable the standby database to be open for normal operations with both read-only and read / write accessibility
- Enable additional objects to be built and maintained

- Enable rolling database upgrades of the production database

A recommended configuration for many cases includes both physical and logical standby databases. They can reside on the same database machine or cluster, but they should be remote from the production database. The physical standby database can be reserved for failovers in case of disaster, and the logical standby database can continue to be used for reporting. The physical standby database provides a faster apply technology because redo logs do not have to be converted to SQL.

Figure 4–3 shows the production database at the primary site and the standby databases at the secondary site.

Figure 4–3 "Data Guard Only" Architecture on Primary and Secondary Sites



See Also:

- *Oracle Data Guard Concepts and Administration* for more information about datatypes supported by logical standby databases
- The papers about standby databases at `http://otn.oracle.com/deploy/availability/htdocs/maa.htm`

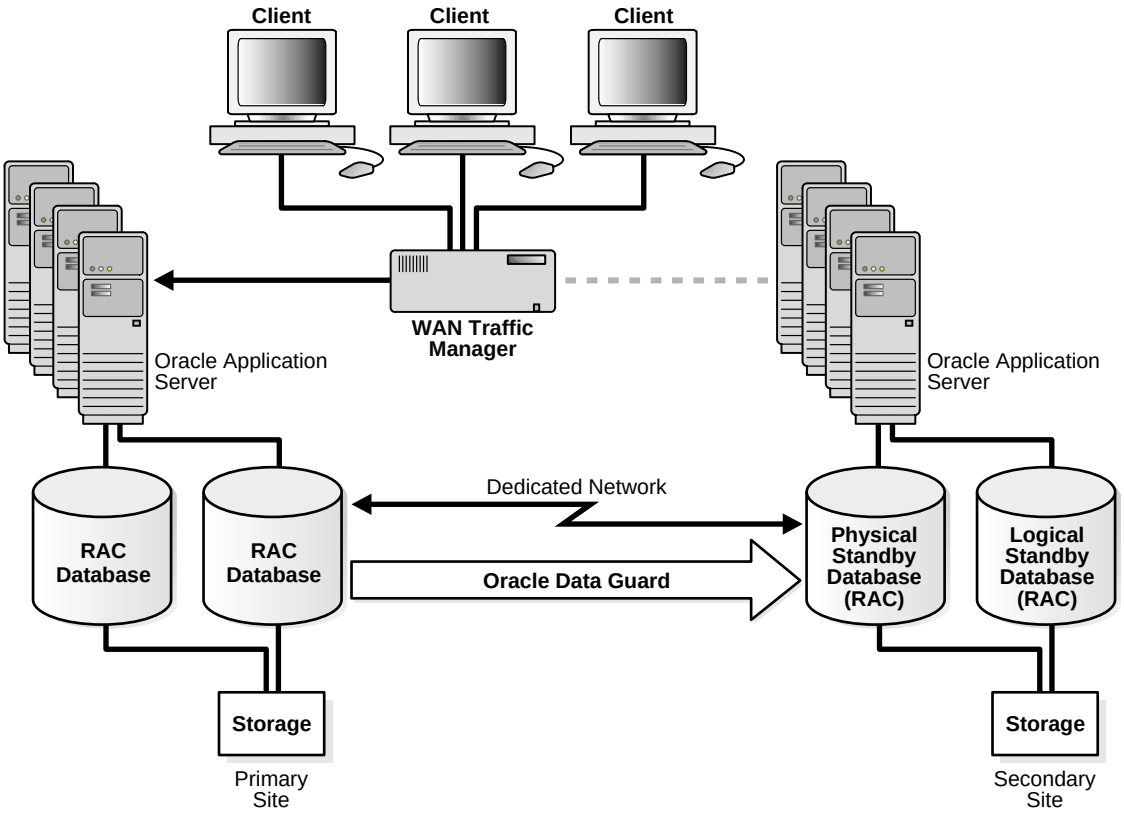
Maximum Availability Architecture

RAC and Data Guard provide the basis of the database MAA solution. MAA provides the most comprehensive architecture for reducing downtime for scheduled outages and preventing, detecting, and recovering from unscheduled outages. The recommended MAA has two identical sites. The primary site contains the RAC database, and the secondary site contains both a physical standby database and a logical standby database on RAC.

Identical site configuration is recommended to ensure that performance is not sacrificed after a failover or switchover. Symmetric sites also enable processes and procedures to be kept the same between sites, making operational tasks easier to maintain and execute.

Figure 4–4 provides an overview of the architecture.

Figure 4–4 Maximum Availability Architecture



Streams Architecture

Streams is meant for information sharing and distribution. It can also provide an efficient and highly available architecture.

Streams provides granularity and control over what is replicated and how it is replicated. It supports bidirectional replication, data transformations, subsetting, custom apply functions, and heterogeneous platforms. It also gives users complete control over the routing of change records from the primary database to a replica database. This enables users to build hub and spoke network configurations that can support hundreds of replica databases.

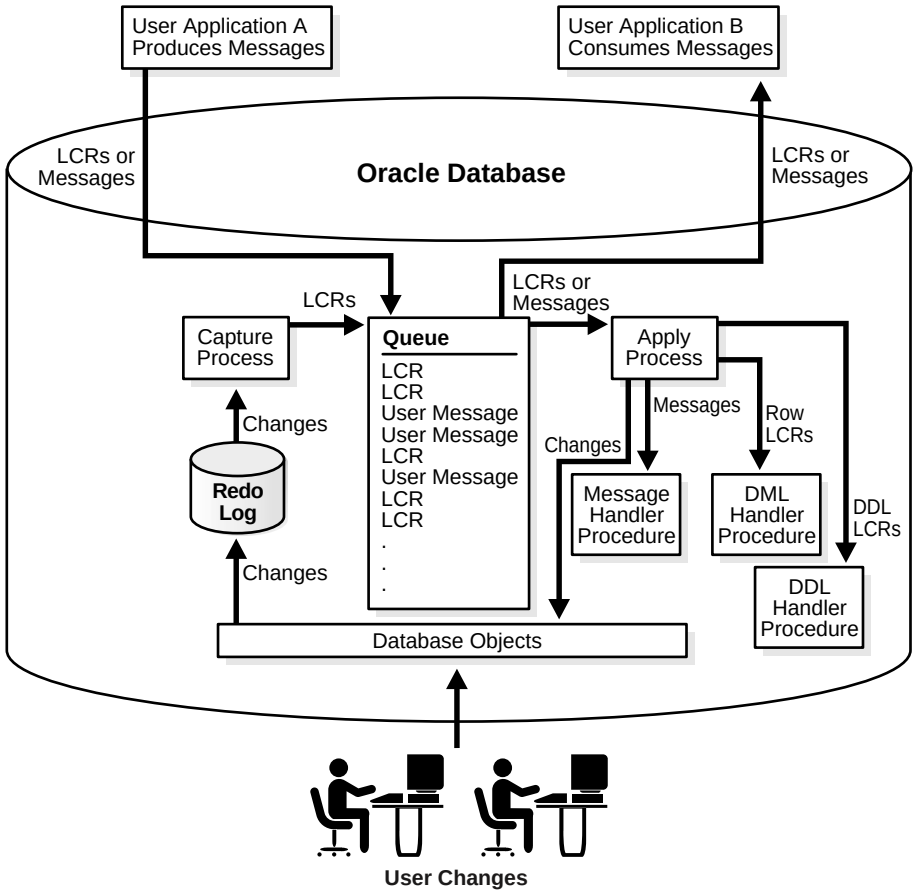
Streams should be evaluated if one or more of the following conditions are true:

- A full active / active site configuration is required including bidirectional changes
- Site configurations are on heterogeneous platforms
- Fine control of information and data sharing are required
- More investment to build an integrated HA solution is available

For disaster recovery, Data Guard is Oracle's recommended disaster recovery solution.

Figure 4–5 shows Streams architecture.

Figure 4–5 Using Streams in an HA Environment



Choosing the Correct HA Architecture

This section summarizes the advantages of the HA architectures discusses in this chapter and then provides guidelines for you to choose the correct HA architecture for your business.

Table 4–2 summarizes the advantages of the five basic HA architectures described in this chapter.

Table 4–2 HA Architecture Advantages

Database HA Architecture	Advantages
Database only Production database on a standalone machine	Reduced downtime for application and Oracle upgrades Flashback capabilities to protect from user errors, and logical corruptions Online redefinition and reconfiguration to minimize scheduled downtime Backup and recovery capabilities Fast and efficient object re-creation features Simple and integrated management framework with Oracle Enterprise Manager **Note:** All of these advantages are available for the other architectures described in this table, but they will not be repeated.
RAC only RAC production database	RAC HA advantages Transparent to the application **See Also:** ""RAC Only" Architecture" on page 4-5
Data Guard only Production database (not RAC) on the primary site Standby databases on the secondary or disaster recovery site	Data Guard HA advantages Transparent to the application **See Also:** ""Data Guard Only" Architecture" on page 4-7
MAA RAC production database on the primary site Standby databases on the secondary or disaster recovery site	RAC HA advantages Data Guard HA advantages Transparent to the application **See Also:** ""RAC Only" Architecture" on page 4-5 and ""Data Guard Only" Architecture" on page 4-7
Streams Primary site: Production database Secondary or additional sites: Database replicated using Streams	Full active / active configuration Database configurations can be heterogeneous platforms **See Also:** "Streams Architecture" on page 4-10

"RAC only" and "Data Guard only" are the most common Oracle HA architectures, and each provides very significant HA advantages. MAA provides the most redundant and robust HA architecture. It prevents, detects, and recovers from different outages within a small mean time to recover (MTTR), as well as preventing or minimizing downtime for maintenance. The Streams architecture is an

alternative high availability solution, but it requires more customization and may not be as transparent to the application.

You must consider your service level agreements in your decision about which HA architecture to implement. To decide whether to implement an Oracle HA architecture that includes RAC or Data Guard, consider the following questions about your service level agreements:

1. Does your production system need to be available 24 hours each day, 7 days each week, 365 days each year?
2. Does your scheduled maintenance currently exceed your service levels?
3. Is disaster recovery required?

Table 4–3 shows the recommended architecture for each set of answers to the questions.

Table 4–3 Service Level Agreements and Recommended HA Architectures

Answer 1 (Local Site HA)	Answer 2 (Rolling Maintenance)	Answer 3 (Disaster Recovery)	Recommended Architecture
No	No	No	Database only
Yes	No	No	RAC only
No	Yes	No	Data Guard only
No	No	Yes	Data Guard only
Yes	Yes	No	MAA
Yes	No	Yes	MAA
No	Yes	Yes	Data Guard only
Yes	Yes	Yes	MAA

If your business requires any of the following, then you should evaluate Oracle Streams in more depth:

- Full active / active site configuration with bidirectional replication
- Heterogeneous platforms across sites
- Fine control of information and data sharing

Assessing Other Architectures

There are other Oracle and non-Oracle HA solutions. This section focuses on the most common variants. Table 4–4 describes each alternative HA proposal, its disadvantages and the recommended Oracle HA architecture.

Table 4–4 Comparison of HA Architectures

Alternative Architecture	Disadvantages	Recommended Solutions
Primary site: Oracle database on hardware cluster **Note:** This is known as **cold failover**.	No RAC HA capability No Data Guard capability	**1.** RAC **2.** MAA
Primary site: Production database Secondary site: Remote mirrored database	No RAC HA capability No Data Guard capability High network utilization No Oracle switchover and failover integration Customization required Remote mirroring solution must be part of the Oracle Storage Compatibility Program (OSCP)	**1.** Data Guard **2.** MAA
Primary site: RAC production database (RAC geo-cluster) Secondary site: At least one node of RAC geo-cluster	Performance impact because of latency between nodes No disaster recovery protection for data or media failures or database corruptions because there are no separate standby databases	**1.** Data Guard or Streams **2.** MAA
Primary site: RAC production database and local Data Guard	No protection from site disasters and site failures	MAA
Primary site: Production database and local Data Guard	No RAC HA capabilities No protection from disasters and site failures	MAA

5

Operational Policies for High Availability

This chapter describes the policies and procedures that essential for maintaining high availability.

- Introduction to Operational Policies for High Availability
- Service Level Management for High Availability
- Planning Capacity to Promote High Availability
- Change Management for High Availability
- Backup and Recovery Planning for High Availability
- Disaster Recovery Planning
- Planning Scheduled Outages
- Staff Training for High Availability
- Documentation as a Means of Maintaining High Availability
- Physical Security Policies and Procedures for High Availability

Introduction to Operational Policies for High Availability

Operational policies together with service management are fundamental to avoiding and minimizing outages, as well as reducing the time to recover from an outage. Operational policies are the foundation for managing the information technology infrastructure. They focus on process, policy, and management.

Operational policies for high availability focus on setting and establishing processes, policies, and management. They are divided into the following categories:

- Service Level Management for High Availability
- Planning Capacity to Promote High Availability
- Change Management for High Availability
- Backup and Recovery Planning for High Availability
- Disaster Recovery Planning
- Planning Scheduled Outages
- Staff Training for High Availability
- Documentation as a Means of Maintaining High Availability
- Physical Security Policies and Procedures for High Availability

> **See Also:** Chapter 6, "System and Network Configuration" for information about technical best practices

Service Level Management for High Availability

Information Technology (IT) departments are required to deliver increasing levels of service and availability while reducing costs. Service level management is an accepted method to ensure that IT services are meeting the business requirements. Service level management requires a dialogue between IT managers and the company's lines of business. It starts with mapping business requirements to IT investments.

Service level management encompasses complete end-to-end management of the service infrastructure. The foundation of service level management is the service level agreement (SLA). The SLA is critical for building accountability into the provider-client relationship and for evaluating the provider's performance. SLAs are becoming more accepted and necessary as a monitoring and control instrument for the relationship between a customer and the IT supplier (external or internal).

SLAs are developed for critical business processes and application systems, such as order processing. The business individuals who specify the functionality of systems should develop the SLA for those systems. The SLA represents a detailed, complete description of the services that the supplier is obligated to deliver, and the responsibilities of the users of that service. Developing an SLA challenges the client to rank the requirements and focus resources toward the most important requirements. An SLA should evolve with the business requirements.

There is no standardized SLA that meets the needs of all companies, but a typical SLA should contain the following sections:

- Definition of the service provided, the parties involved and the effective dates of the agreement
- Specification of the hours and days during which the service or application will be available, excluding time for scheduled testing, maintenance, and upgrades
- Specifications of the numbers and locations of users and hardware for which the service or application will be offered.
- Problem reporting and escalation procedures, including an expected response time for problems
- Procedures for requesting changes, possibly including expected times for completing routine requests
- Specification of quality targets and explanations of how they are measured and how frequently they are reported
- Specifications of service costs and charges; the charges may be a flat rate or they may be tied to different levels of service quality
- Specifications of user responsibilities, such as user training, maintaining proper desktop configuration, not introducing extraneous software or circumventing change management procedures
- Description of procedures for resolving service-related disagreements

Developing an SLA and service level measurements requires commitment and hard work by all parties. Service levels should be measured by business requirements, such as cost for each order processed. Any shared services or components must perform at the level of the most stringent SLA. Furthermore, SLAs should be developed between interdependent IT groups and with external suppliers. Many technologists advocate an integrated, comprehensive SLA rather than individual SLAs for infrastructure components.

The benefits of developing a SLA are:

- A professional relationship between the supplier and customer with documented accountability
- A mutual goal of understanding and meeting the business requirements
- A system of measurement for service delivery so that IT can quantify their capabilities and results and continuously improve upon them
- IT can prevent or respond faster to events that decrease availability
- A documented set of communication and escalation procedures

Planning Capacity to Promote High Availability

Planning capacity and monitoring thresholds is essential to prevent downtime or unacceptably delayed transactions. Understanding average and maximum usage and the requirements to maintain that load over time helps ensure acceptable performance.

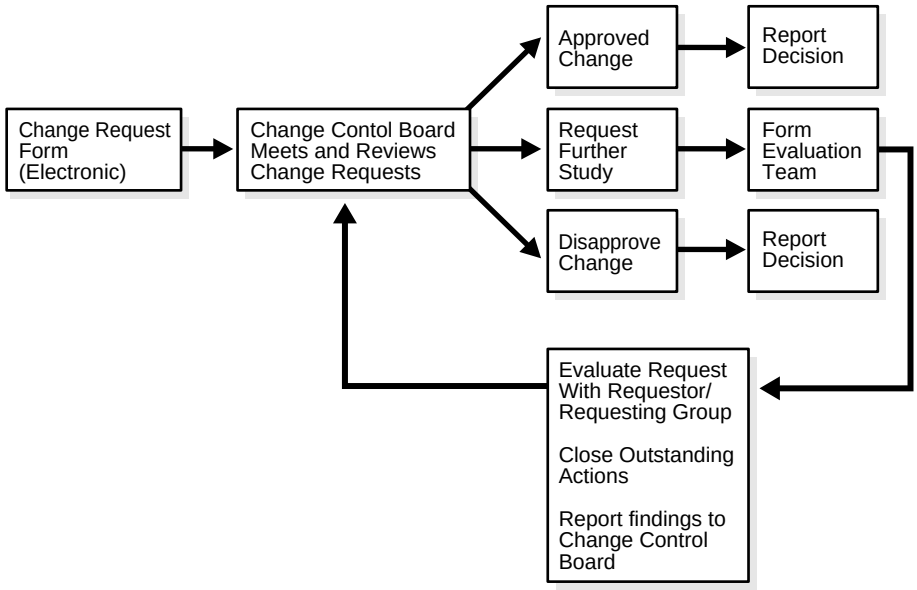
Capacity planning includes the ability to estimate the time remaining before a tablespace becomes completely full and planning ahead to add disk space. Capacity planning can also delay or prevent scheduled outages to increase the maximum number of sessions in the database.

Change Management for High Availability

Change management is a set of procedures or rules that ensure that changes to the hardware, software, application, and data on a system are authorized, scheduled, and tested. A stable system in which unexpected, untested, and unauthorized changes are not permitted is one that guarantees integrity to its users. The users can rely on the hardware, software, and data to perform as anticipated. Knowing exactly when and what changes have been applied to the system is vital to debugging a problem. Each customer handles change management of systems, databases, and application code differently, but there are general guidelines that can help prevent unnecessary system outages, thus protecting the system's integrity. With proper change management, application and hardware systems have greater stability, and new problems are easier to debug.

Figure 5–1 describes a typical change control flow. For emergency cases such as disasters, the change control process may need to be shortened.

Figure 5–1 Change Control



The following recommendations are the foundation of good change management:

- Develop a change control process

  A change control process for both nonescalated and escalated cases should be created, documented, and implemented. Ad hoc and emergency changes to the hardware, database, and software in a system are inevitable, but the change control process must ensure that they are later incorporated into the change management system so their effects and ramifications are examined and recorded.

- Form a change control group

  Representatives from applications, databases, systems, and management should be members of the change control board. Both hardware and software representatives must be present.

  Determine meeting frequency and minimum assessment time. Change control processes should allow essential changes to be implemented in a reasonable time. Change control meetings should be frequent enough to address and discuss the most important issues. There should be a minimum grace period from the time a change is submitted until the time it is scheduled for review to

provide adequate assessment time. This assessment time should be bypassed only with upper management approval.

- Evaluate proposed changes

  Changes must provide short-term or long-term benefit. The change control team needs to assess whether the benefits of a change outweigh the risks and whether the change is consistent with the overall vision of the business, the application, and its rules. Proposed change must document the following:

  - Purpose of the change
  - Risk assessment
  - Fall-back plans
  - Test plans and results of the tests
  - Estimated time to implement and back out change, including outage times

- Gather statistics for base comparisons

  Gather snapshots of system, hardware, database, and application configuration and performance statistics. These base numbers can be used for comparison when a change is implemented. After changing a database parameter, you can gather new statistics and compare them with the base statistics to determine the impact of a change.

- Track database changes

  Database structure changes are easy to make on demand and easy to slip through the change management process. Therefore, it may be necessary to have a special procedure in place for these changes. It is also beneficial to track these changes for trend analysis. In addition, when files are added to the database, the files must be incorporated into the backup and monitoring schemes; proper tracking of this type of change can act as a reminder.

- Use a version control system for application code

  Some version control system must exist for application code to help track changes and enable fallback to a previous code base. Internally developed applications are modified and enhanced frequently, and new versions are put in place. When a problem is found with the new version, testing the case in the old version provides valuable debugging information. Depending on the type of application and the likelihood of the users' need to revert to an earlier version, the company must decide how many previous versions to keep on hand. At least one is mandatory.

- Develop quality assurance and testing procedures

  Quality assurance should validate test specifications, test plans, and the results of tests to ensure that the test simulation mimics your applications or at least considers all critical points of the application being tested. Tests and test environments should be designed to test both essential functionality and scalability of the application.

- Perform internal audits

  Internal audits may be used to verify that your hardware, software, database and application are certified with vendors, performing within service levels, and achieving high availability.

Backup and Recovery Planning for High Availability

Proper backup and recovery plans are essential and must be constructed to meet your specific service levels. Both disk and tape database backups are recommended. Disk and tape backups are essential for disaster recovery and for cases when you need to restore from a very old backup.

A robust backup and recovery scheme requires an understanding of how it is strengthened or compromised by the physical location of files, the order of events during the backup process, and the handling of errors. A robust scheme is resilient in the face of media failures, programmatic failures, and, if possible, operator failures. Complete, successful, and tested processes are fundamental to the successful recovery of any failed environment.

Take the following steps to construct useful backup and recovery plans:

- Create recovery plans

  Create recovery plans for different types of outages. Start with the most common outage types and progress to the least probable. An outage matrix with recommended recovery actions and a validated MTTR estimate enables you to assess if you can meet your SLAs for different types of outages.

  See Also: Chapter 9, "Recovering from Outages"

- Test backups on a regular basis

  Monitor the backup tasks for errors and validate backups by testing your recovery procedures periodically.

- Automate backup and recovery procedures

- Choose an appropriate backup frequency

  Having up-to-date backups reduces recovery time.

- Maintain offsite tape backups

  Offsite backups of the database are essential to protect from site failures.

- Maintain updated documentation for backup and recovery plans

  Documentation is, for obvious reasons, a safeguard against mistakes, loss of knowledgeable people, and misinterpretations. Maintaining accurate documentation on backup and recovery procedures is as important as having procedures in place.

Disaster Recovery Planning

Disaster recovery (DR) planning is a process designed and developed specifically to deal with catastrophic, large-scale interruptions in service to allow timely resumption of operations. These interruptions can be caused by disasters like fire, flood, earthquakes, or malicious attacks. The basic assumption is that the building where the data center and computers reside may not be accessible, and that the operations need to resume elsewhere. It assumes the worst and tries to deal with the worst. As an organization increasingly relies on its electronic systems and data, access to these systems and data become a fundamental component of success. This underscores the importance of disaster recovery planning. Proper disaster planning reduces MTTR during a catastrophe and provides continual availability of critical applications, helping to preserve customers and revenue.

Take the following steps to plan recovery from disasters:

- Choose the right disaster recovery plans (DRPs)

  DRPs must deliver the expected MTTR service levels. The implementation costs must also be justified by the service levels. One DRP may not accommodate all disasters or even the common disasters.

- Determine what is covered under the disaster recovery plans

  The first question to ask when trying to decide whether to include an application in the disaster recovery plans is whether that application supports a key business operation that must be brought back online within a few hours or days of a disaster. This may not be the same as the availability requirements of the application, although it is closely related. It has more to do with the cost to the company every hour or day that the system is not available. Disaster recovery planning requires securing off-site equipment, personnel, and

supporting components such as phone lines and networks that can function at an acceptable level in an interim basis. This is costly, and care must be taken to consider only those applications that are key to the survival of the company.

- Document DRPs, including diagrams of affected areas and systems

  A DRP should clearly identify the outage it protects against and the steps to implement in case of that outage. A general diagram of the system is essential. It needs to be detailed enough to determine hardware fault tolerance. including controllers, mirrored disks, the disaster backup site, processors, communication lines, and power. It also helps identify the current resources and any additional resources that may be necessary. Understanding how and when data flows in and out of the system is essential in identifying parts of the system that merit special attention. Special attention can be in the form of additional monitoring requirements or the frequency and types of backups taken. Conversely, it may also show areas that only require minimal attention and fewer system resources to monitor and manage.

- Set up disaster recovery processes

  Ensure that critical applications, database instances, systems, or business processes are included in your disaster recovery plan. Use application, system and network diagrams to assess fault tolerance and alternative routes during a disaster.

- Assess all of the important business components

  Consider all the components that allow your business to run. Ensure that the DRP includes all system, hardware, application and people resources. Verify that network, telephone service and security measures are in place.

- Assign a DR coordinator

  A DR coordinator and a backup coordinator should be pre-assigned to ensure that all operations and communications are passed on.

- Test and validate the DRP

  The DRP must be rehearsed periodically, which implies that the facilities to test the DRP must be available.

Planning Scheduled Outages

Scheduled outages can affect the application server tier, the database tier, or the entire site. These outages may include one or more of the following: node hardware maintenance, node software maintenance, Oracle software maintenance, redundant

component maintenance, entire site maintenance. Proper scheduled outage planning reduces MTTR and reduces risk when changes do not go as planned.

Take the following steps to plan scheduled outages:

- Create a list of scheduled outages

  Creating a list of possible scheduled outages, their projected frequency, and estimated duration enables advanced planning and a better assessment of availability requirements. A reliability assessment to understand the mean time between failures (MTBF) of critical components can be used to plan for certain scheduled outages in order to prevent an unscheduled outage. In many cases, only one large scheduled outage is allotted each year, so maintenance must be prioritized and justified.

- For each possible scheduled outage, document the impact and assess the risk

  Scheduled outages that do not require software or application changes can usually be done with minimum downtime if a subsequent system can take over the new transactions. With Real Application Clusters and Data Guard switchover, you can upgrade hardware and do some standard system maintenance with minimum downtime to your business. For most software upgrades such as Oracle upgrades, the actual downtime can be less than an hour if prepared correctly. For more complex application changes that require schema changes or database object reorganizations, customers must assess whether Oracle's online reorganization features suffice or use some of Oracle's rolling upgrade capabilities.

  See Also: "Recovery Steps for Scheduled Outages" on page 9-7

- Justify the scheduled outage

  Each change must be consistent with the overall vision of the application and business and adhere to compatibility and change control rules.

- Create and automate change, testing, and fallback procedures

  Each planned change, such as an Oracle upgrade, should be tested in a simulated real world environment to assess performance and availability impacts. Oracle recommends using complete stress tests and a load simulated to accurately assess performance and load impact. Fallback plans should be created and incorporated into the scheduled outage. An automated process should be in place to implement the change and properly fall back if required.

Staff Training for High Availability

Highly trained people can make better and more informed decisions and are less likely to make mistakes. A comprehensive plan for the continued technical education of the systems administration, database administration, development, and users groups can help ensure higher availability of your systems and databases. Additionally, just as redundancy of system components eliminates a single point of failure, knowledge management and cross training should eradicate the effects to operations of losing an employee.

- Cross-train for business-critical positions

  Any business-critical systems should have cross-training of technical staff to reduce the impact to operations if an employee becomes unavailable or leaves the company. For example, the system administration group should be familiar with Oracle RDBMS and tools. Maintain formal and regular forms of communication (such as weekly meetings) between different support groups.

- Develop guidelines to ensure continued technical education

  Ensure that there is a process in place to notify and train staff about new features or procedures associated with the hardware and software your company uses. Additionally, allow time for investigation into new technologies that can improve service levels or reduce costs.

- Implement a knowledge management process

  Effectively managing the intellectual assets of a company reduces the risk of losing those assets. Create a process to promote central access to information about "lessons learned" within the IT group. For example, group round tables, internal white papers, new features related to upgrades, repositories for problem analysis and resolutions are ways of making information accessible.

- Update training materials when applications or system are changed

  Training material should be kept up to date with application and system changes. Incorporate training materials into the change management and documentation procedures.

Documentation as a Means of Maintaining High Availability

Clear and complete documentation should be part of every set of HA operational policies. Without documenting the steps for implementing or executing a process, you run the risk of losing that knowledge, increasing the risk for human error

during the execution of that process, and omitting a step within a process. All of these risks affect availability.

Clearly defined operational procedures contribute to shorter learning curves when new employees join your organization. Properly documented operational procedures can greatly reduce the number of questions for support personnel, especially when the people who put the procedures in place are no longer with the group. Proper documentation can also eliminate confusion by clearly defining roles and responsibilities within the organization.

Clear documentation of applications is essential for new employees. When internally developed applications need to be maintained and enhanced, documentation helps developers refresh their knowledge of the internal details of the programs. If the original developers are no longer with the group, this documentation becomes especially valuable to new developers who would otherwise have to struggle through reading the code itself. Readily available application documentation also can greatly reduce the number of questions for your support organization.

- Ensure that documentation is kept up to date

  Update operational procedures when an application or system changes. Keep users informed of documentation updates.

- Approve documentation changes through the change management process

  The change management team should review and approve changes to the documentation to ensure accuracy.

- Document lessons learned and problem resolutions

  Documenting problem resolutions and lessons learned can improve the recovery time for repeated problems. Ideally, this documentation can be part of a periodic review process to help set priorities for system enhancements.

- Protect the documentation

  Secure access to documentation and keep an offsite copy of your operational procedure documentation and any other critical documentation. All critical documentation should also be part of any remote site implementations. Whether the remote site is intended for restarting a system after a disaster or for disaster recovery, the site should also contain a copy of the documented procedures for failing over to that site.

Physical Security Policies and Procedures for High Availability

Security policies consider the physical security and operations of the hardware and the data center. Physical security includes protection from unauthorized access, as well as from physical damage such as from fire, heat, and electrical surges. Physical security is the most fundamental security precaution and is essential for the system to meet the customer's availability requirements. Physical security protects against external and internal security breach. The CSI/FBI Computer Crime and Security Survey documents a trend toward increasing external intrusions and maintains that internal security violations still pose a large threat. A detailed discovery process into the security of data center operations and organization is outside the scope of this book. However, a properly secured infrastructure reduces the risk of damage, downtime, and financial loss.

Take the following steps to maintain physical security of the hardware and data center:

- Provide a suitable physical environment for computer equipment

  Not every room or closet in an office environment can be used to house computer equipment. The data center should not only account for the appropriate temperature, humidity, and security of the systems, it should also attempt to prevent potential hazards such as electrical surge, fire, and flood.

- Restrict access to the operations area to authorized personnel

  All security-conscious operations centers need to have some sort of secure access, either in the form of biometric authentication devices or smart-card readers.

- Use internal security monitoring

  Devices such as cameras and closed-circuit television are essential to a secure operations center by preventing crime and damage caused by people who are inside the facility.

- Conduct background checks on DBAs, system administrators, and operational staff

  DBAs, system administrators, and operational staff are inherently privileged users and hold positions of trust. Organizations must perform adequate background checks to ensure that these privileged individuals are worthy of the trust placed in them. There is no technical solution that can completely protect against a determined, malicious, and poorly evaluated person holding a position of power.

See Also: Chapter 6, "System and Network Configuration" for information about data security

Part III

Configuring a Highly Available Oracle Environment

This part describes how to configure the high availability architectures.

This part contains the following chapters:

- Chapter 6, "System and Network Configuration"
- Chapter 7, "Oracle Configuration Best Practices"

6

System and Network Configuration

This chapter provides recommendations for configuring the subcomponents that make up the database server tier and the network. It includes the following sections:

- Overview of System Configuration Recommendations
- Recommendations for Configuring Storage
- Recommendations for Configuring Server Hardware
- Recommendations for Configuring Server Software
- Recommendations for Configuring the Network

Overview of System Configuration Recommendations

The goal of configuring a highly available environment is to create a redundant, reliable system and database without sacrificing simplicity and performance. This chapter includes recommendations for configuring the subcomponents of the database server tier.

These principles apply to all of the subcomponent recommendations:

- **Redundancy**: Choose components and capabilities that are redundant to reduce the risk from failures
- **Consolidation**: Consolidate components where appropriate to reduce the number of components that can fail
- **Expansion**: Plan for future growth
- **Manageability**: Choose components that are reliable and can be serviced online

> **See Also:** Your vendors' HA architecture and best practices documentation and recommendations for hardware, operating system, and cluster configurations

Recommendations for Configuring Storage

Electronic data is one of the most important assets of any business. Storage arrays that house this data must protect it and keep it accessible to ensure the success of the company. This section describes characteristics of a fault-tolerant storage subsystem that protects data while providing manageability and performance. The following storage recommendations for all architectures are discussed in this section:

- Ensure That All Hardware Components Are Fully Redundant and Fault-Tolerant
- Use an Array That Can Be Serviced Online
- Mirror and Stripe for Protection and Performance
- Load-Balance Across All Physical Interfaces
- Create Independent Storage Areas
- Define ASM Disk and Failure Groups Properly
- Use HARD-Compliant Storage for the Greatest Protection Against Data Corruption

The following section pertains specifically to RAC environments:

- Storage Recommendation for RAC

Ensure That All Hardware Components Are Fully Redundant and Fault-Tolerant

All hardware components of a storage array must be fully redundant, from physical interfaces to physical disks, including redundant power supplies and connectivity to the array itself.

The storage array should contain one or more spare disks (often called hot spares). When a physical disk starts to report errors to the monitoring infrastructure, or fails suddenly, the firmware should immediately restore fault tolerance by mirroring the contents of the failed disk onto a spare disk.

Connectivity to the storage array must be fully redundant (referred to as multipathing) so that the failure of any single component in the data path from any node to the shared disk array (such as controllers, interface cards, cables, and switches) is transparent and keeps the array fully accessible. This achieves addressing the same logical device through multiple physical paths. A host-based device driver reissues the I/O to one of the surviving physical interfaces.

If the storage array includes a write cache, then it must be protected to guard against memory board failures. The write cache should be protected by multiple battery backups to guard against a failure of all external power supplies. The cache must be able to survive either by using the battery backup long enough for the external power to return or until all the dirty blocks in cache are guaranteed to be fully flushed to a physical disk.

Use an Array That Can Be Serviced Online

If a physical component fails, then the array must allow the failed device to be repaired or replaced without requiring the array to be shut down or taken offline. Also, the storage array must allow the firmware to be upgraded and patched without shutting down the storage array.

Mirror and Stripe for Protection and Performance

Data should be mirrored to protect against disk and other component failures and should be striped over a large number of disks to achieve optimal performance. This method of storage configuration is known as Stripe and Mirror Everything (SAME). The SAME methodology provides a simple, efficient, and highly available storage configuration.

Oracle's automatic storage management (ASM) feature always evenly stripes data across all drives within a disk troup with the added benefit of automatically rebalancing files across new disks when disks are added, or across existing disks if disks are removed. In addition, ASM can provide redundancy protection to protect against component failures or enable mirroring to be provided by the underlying storage array.

See Also:

- "Optimal Storage Configuration Made Easy" at `http://otn.oracle.com/deploy/availability/pdf/oow2000_same.pdf`
- An internal Oracle study validated the assertions made in "Optimal Storage Configuration Made Easy". The study can be found at `http://otn.oracle.com/deploy/availability/pdf/SAME_HP_WP_112002.pdf`
- *Oracle Database Administrator's Guide* for more information about ASM

Load-Balance Across All Physical Interfaces

Load balancing of I/O operations across physical interfaces is usually provided by a software package that is installed on the host. The purpose of this load balancing software is to redirect I/O operations to a less busy physical interface if a single host bus adapter (HBA) is overloaded by the current workload.

Create Independent Storage Areas

Create separate, independent storage areas for software, active database files, and recovery-related files.

The following storage areas are needed:

- Software area: A location where software is installed and log and trace files are created
- Database area: A location where active database files such as datafiles, control files, online redo logs, and change tracking files used in incremental backups are stored

- Flash recovery area: A location where recovery-related files are created, such as multiplexed copies of the current control file and online redo logs, archived redo logs, backup sets, and flashback log files

The storage containing the database area should be as physically distinct as possible from the flash recovery area. At a minimum, the database area and flash recovery area should not share the same physical disk drives or controllers. This practice ensures that the failure of a component that provides access to a datafile in the database area does not also cause the loss of the backups or redo logs in the flash recovery area needed to recover that datafile.

Storage options are defined as follows:

- Regular file system: A local file system that is not shared by any other system
- Cluster file system (CFS): A file system that is equally addressable and accessible by all nodes in a cluster (unlike a shared file system where access is funneled through a master node)
- Automatic storage management (ASM): A tool designed specifically for Oracle database files that integrates the capabilities of a file system and a volume manager
- Raw device or logical volume: A piece of storage typically managed by a logical volume manager that does not contain a file system

The rest of this section includes these topics:

- Storage Recommendations for Specific HA Architectures
- Define ASM Disk and Failure Groups Properly

Storage Recommendations for Specific HA Architectures

For the "Data Guard only" architecture and MAA, the primary site and secondary site should each contain their own identically configured storage areas.

For the "RAC only" architecture and MAA, follow these recommendations:

- Software area should be installed on a regular file system local to each node in the RAC cluster. This configuration permits Oracle patch upgrades and eliminates the software as a single point of failure.
- Both the database area and the flash recovery area must be accessible to all nodes in the RAC cluster.

- If a cluster file system (CFS) is available on the target platform, then both the database area and flash recovery area can be created on either CFS or ASM.
- If a cluster file system (CFS) is unavailable on the target platform, then the database area can be created either on ASM or on raw devices (with the required volume manager), and the flash recovery area must be created on ASM.

Table 6–1 summarizes the independent storage recommendations by architecture.

Table 6–1 Independent Storage Recommendations for HA Architectures

Storage Area	Non-RAC Database	RAC Database (CFS Available)	RAC Database (CFS Not Available)
Software area	Regular file system	Regular file system	Regular file system
Database area	Regular file system or ASM	CFS or ASM	Raw or ASM
Flash recovery area	Regular file system or ASM	CFS or ASM	ASM

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide* for details about moving an existing database into ASM
- Your platform-specific installation guide and *Oracle Real Application Clusters Installation and Configuration Guide* for details about setting up disks for use with ASM

Define ASM Disk and Failure Groups Properly

ASM uses a concept called failure groups to protect data against disk or component failures. When using failure groups, ASM optimizes file layout to reduce the unavailability of data due to the failure of a shared resource. Failure groups define disks that share components, so that if one disk fails, then other disks sharing the component might also fail. An example of what might be defined as a failure group is a string of SCSI disks on the same SCSI controller. Failure groups are used to determine which ASM disks to use for storing redundant data. For example, if 2-way mirroring is specified for a file, then redundant copies of file extents will be stored in separate failure groups.

Failure groups are used with storage that does not provide its own redundancy capability, such as disks that have not been configured according to RAID. The manner in which failure groups are defined for an ASM disk group is site-specific

because the definition depends on the configuration of the storage and how it is connected to the database systems. ASM attempts to maintain three copies of its metadata, requiring a minimum of three failure groups for proper protection.

When using ASM with intelligent storage arrays, the storage array's protection features (such as hardware RAID-1 mirroring) are typically used instead of ASM's redundancy capabilities, which are implemented by using the `EXTERNAL REDUNDANCY` clause of the `CREATE DISKGROUP` statement. When using external redundancy, ASM failure groups are not used, because disks in an external redundancy disk group are presumed to be highly available.

Disks, as presented by the storage array to the operating system, should not be aggregated or subdivided by the array because it can hide the physical disk boundaries needed for proper data placement and protection. If, however, physical disk boundaries are always hidden by the array, and if each logical device, as presented to the operating system, has the same size and performance characteristics, then simply place each logical device in the same ASM disk group (with redundancy defined as `EXTERNAL REDUNDANCY`),and place the database and flash recovery areas in that one ASM disk group. An alternative approach is to create two disk groups, each concsisting of a single logical device, and place the database area in one disk group and the flash recovery in the other disk group. This method provides additional protection against disk group metadata failure and corruption.

For example, suppose a storage array takes eight physical disks of size 36GB and configures them in a RAID 0+1 manner for performance and protection, giving a total of 144GB of mirrored storage. Futhermore, this 144GB of storage is presented to the operating system as two 72GB logical devices with each logical device being striped and mirrored across all eight drives. When configuring ASM, place each 72GB logical device in the same ASM disk group, and place the database area and flash recovery areas on that disk group.

If the two logical devices have different performance characteristics,(for example, one corresponds to the inner half and the other to the outer half of the underlying physical drives) then the logical devices should be placed in separate disk groups. Because the outer portion of a disk has a higher transfer rate, the outer half disk group should be used for the database area; the inner half disk group should be used for the flash recovery area.

If multiple, distinct storage arrays are used with a database under ASM control, then multiple options are available. One option is to create multiple ASM disk groups that do not share storage across multiple arrays and place the database and flash recovery areas in separate disk groups, thus physically separating the database area from the flash recovery area. Another option is to create a single disk

group across arrays that consists of failure groups, where each failure group contains disks from just one array.These options provide protection against the failure of an entire storage array.

See Also: *Oracle Database Administrator's Guide*

Use HARD-Compliant Storage for the Greatest Protection Against Data Corruption

The Hardware Assisted Resilient Data (HARD) initiative is a program designed to prevent data corruptions before they happen. Data corruptions are very rare, but when they do occur, they can have a catastrophic effect on business. Under the HARD initiative, Oracle's storage partners implement Oracle's data validation algorithms inside storage devices. This makes it possible to prevent corrupted data from being written to permanent storage. The goal of HARD is to eliminate a class of failures that the computer industry has so far been powerless to prevent. RAID has gained a wide following in the storage industry by ensuring the physical protection of data; HARD takes data protection to the next level by going beyond protecting physical data to protecting business data.

In order to prevent corruptions before they happen, Oracle tightly integrates with advanced storage devices to create a system that detects and eliminates corruptions before they happen. Oracle has worked with leading storage vendors to implement Oracle's data validation and checking algorithms in the storage devices themselves. The classes of data corruptions that Oracle addresses with HARD include:

- Writes that physically and logically corrupt data file, control file and log file blocks
- Writes of Oracle blocks to incorrect locations
- Erroneous writes to Oracle data by programs other than Oracle
- Writes of partial or incomplete blocks

End-to-end block validation is the technology employed by the operating system or storage subsystem to validate the Oracle data block contents. By validating Oracle data in the storage devices, corruptions will be detected and eliminated before they can be written to permanent storage. This goes beyond the current Oracle block validation features that do not detect a stray, lost, or corrupted write until the next physical read.

Oracle vendors are given the opportunity to implement validation checks based on a specification. A vendors' implementation may offer features specific to their storage technology. Oracle maintains a Web site that will show a comparison of each

vendor's solution by product and Oracle version. For the most recent information, see `http://otn.oracle.com/deploy/availability/htdocs/HARD.html`.

> **See Also:** Appendix A, "Hardware Assisted Resilient Data (HARD) Initiative"

Storage Recommendation for RAC

The following recommendation applies to the "RAC only" architecture and MAA:

- Protect the Oracle Cluster Registry and Voting Disk From Media Failure

Protect the Oracle Cluster Registry and Voting Disk From Media Failure

The shared volumes created for the OCR and the voting disk should be configured using RAID to protect against media failure. This requires the use of an external cluster volume manager, cluster file system, or storage hardware that provides RAID protection.

> **See Also:** *Oracle Real Application Clusters Installation and Configuration Guide*

Recommendations for Configuring Server Hardware

The main server hardware components are the nodes for the database and application server farm and the components within each node such as CPU, memory, interface boards (such as I/O and network), storage, and the cluster interconnect in a RAC environment.

This section includes the following topics:

- Server Hardware Recommendations for All Architectures
- Server Hardware Recommendations for RAC
- Server Hardware Recommendations for Data Guard

Server Hardware Recommendations for All Architectures

The following recommendations apply to all architectures:

- Use Fewer, Faster, and Denser Components
- Use Redundant Hardware Components
- Use Systems That Can Detect and Isolate Failures

- Protect the Boot Disk With a Backup Copy

Use Fewer, Faster, and Denser Components

Use fewer, faster CPUs instead of more, slower CPUs. Use fewer higher-density memory modules instead of more lower-density memory modules. This reduces the number of components that can fail while providing the same service level. This needs to be balanced with cost and redundancy.

Use Redundant Hardware Components

Using redundant hardware components enables the system to fail over to the working component while taking the failed component offline. Choose components (such as power supplies, cooling fans, and interface boards) that can be repaired or replaced while the system is running to prevent unscheduled outages caused by the repair of hardware failures.

Use Systems That Can Detect and Isolate Failures

Use systems that can automatically detect failures and provide alternate paths around subsystems that have failed or isolate the subsystem. Choose a system that continues to run despite a component failure and automatically works around the failed component without incurring a full outage. For example, find a system that can use an alternate path for I/O requests if an adapter fails or can avoid a bad physical memory location if a memory board fails.

Protect the Boot Disk With a Backup Copy

Because mirroring does not protect against accidental removal of a file or most corruptions of the boot disk, an online copy of the boot disk should be maintained so that the system can be quickly rebooted using the same operating system image if a critical file is removed or corruption occurs on the primary boot image. Operating system vendors often provide a mechanism to easily maintain multiple boot images.

Server Hardware Recommendations for RAC

The following recommendations apply to the "RAC only" and MAA environments:

- Use a Supported Cluster System to Run RAC
- Choose the Proper Cluster Interconnect

Use a Supported Cluster System to Run RAC

RAC provides fast, automatic recovery from node and instance failures. Using a properly supported configuration is necessary for a successful RAC implementation. A supported hardware configuration may encompass server hardware, storage arrays, interconnect technology, and other hardware components.

Choose the Proper Cluster Interconnect

Select an interconnect that has redundancy, high speed, low latency, low host resource consumption, and the ability to balance loads across multiple available paths. In two-node configurations, it may be possible to use a direct-connect interconnect between the two nodes, but a switch should be used instead to provide a degree of isolation between the network interface cards of the two nodes in the cluster. If you plan to have more than two nodes in the future, then choose a switch-based interconnect solution instead of direct-connect to reduce the complexity of adding additional nodes in the future. If you have more than two nodes, then a switch-based interconnect is highly recommended and, in many cases, a requirement of the cluster solution being used.

Server Hardware Recommendations for Data Guard

The following recommendation applies to both the primary and secondary sites in "Data Guard only" and MAA environments.

Use Identical Hardware for Every Machine at Both Sites

Using identical hardware for machines at both sites provides a symmetric environment that is easier to administer and maintain. Such a symmetric configuration mitigates failures or performance inconsistencies following a switchover or failover because of dissimilar hardware.

Recommendations for Configuring Server Software

The recommendations for serversoftware apply to all nodes in a RAC environment and to both the primary and secondary sites in a Data Guard and MAA environment because they contain the identical configuration.

This section includes the following topics:

- Server Software Recommendations for All Architectures
- Server Software Recommendations for RAC

Server Software Recommendations for All Architectures

The following recommendations apply to all architectures:

- Use the Same OS Version, Patch Level, Single Patches, and Driver Versions
- Use an Operating System That is Fault-Tolerant to Hardware Failures
- Configure Swap Partitions Appropriately
- Set Operating System Parameters to Enable Future Growth
- Use Logging or Journal File Systems
- Mirror Disks That Contain Oracle and Application Software

Use the Same OS Version, Patch Level, Single Patches, and Driver Versions

Use the same operating system version, patch level, single patches, and driver versions on all machines. Consistency with operating system versions and patch levels reduces the likelihood of encountering incompatibilities or small inconsistencies between the software on all machines. In an open standards environment, it is impractical for any vendor to test each and every piece of new software with every combination of software and hardware that has been previously released. Temporary differences can be tolerated when using RAC or Data Guard as individual systems or groups of systems are upgraded or patched one at a time to minimize scheduled outages, if the goal is that all machines will be upgraded to the same versions and patch levels.

Use an Operating System That is Fault-Tolerant to Hardware Failures

Use an operating system that, when coupled with the proper hardware, supports the ability to automatically detect failures and provide alternate paths around subsystems that have failed or isolate subsystems. Choose a system that can continue running if a component, such as a CPU or memory board, fails and automatically provides paths around failed components, such as an alternate path for I/O requests in the event of an adapter failure.

Configure Swap Partitions Appropriately

Mirror disks containing swap partitions so that if a disk that contains a swap partition fails, it will not result in an application or system failure.

Use disk-based swap instead of RAM-based swap. It is always good practice to make all system memory available for database and application use unless the amount available is sufficiently oversized to accommodate swap.

Do not use TMPFS (a Solaris implementation that stores files in virtual memory rather than on disk) or other file systems that exclusively use virtual memory instead of disks for storage for the `/tmp` file system or other scratch file systems. This prevents runaway programs that write to `/tmp` from having the potential to hang the system by exhausting all virtual memory.

Set Operating System Parameters to Enable Future Growth

An example on UNIX is setting shared memory and semaphore kernel parameters high enough to enable future growth, thus preventing an outage for reconfiguring the kernel and rebooting the system. Verify that using settings higher than required either presents no overhead or incurs such small overhead on large systems that the effect is negligible.

Use Logging or Journal File Systems

Journal file systems and logging reduce or eliminate the number of file system checks required following a system reboot, thereby facilitating faster restarting of the system.

Mirror Disks That Contain Oracle and Application Software

As with server hardware and software, the recommendations for Oracle software apply to both the primary and secondary sites because they contain identical configurations. Mirror disks containing Oracle and application software to prevent a disk failure from causing an unscheduled outage.

Server Software Recommendations for RAC

The following recommendations apply to a "RAC only" environment:

- Use Supported Clustering Software
- Use Network Time Protocol (NTP) On All Cluster Nodes

Use Supported Clustering Software

RAC provides fast, automatic recovery from node and instance failures. Using a properly supported configuration is a key component of success. A supported software configuration encompasses operating system versions and patch levels, clustering software versions, which possibly includes Oracle-supplied cluster software. On some platforms (such as Solaris, Linux, and Windows), Oracle supplies cluster software required for use with RAC.

See Also: *Oracle Real Application Clusters Installation and Configuration Guide*

Use Network Time Protocol (NTP) On All Cluster Nodes

Use NTP to synchronize the clocks on all nodes in the cluster to facilitate analysis of tracing information based on timestamps.

Recommendations for Configuring the Network

This section includes the following topics:

- Network Configuration Best Practices for All Architectures
- Network Configuration Best Practices for RAC
- Network Configuration Best Practices for Data Guard

Network Configuration Best Practices for All Architectures

The following recommendations apply to all architectures:

- Ensure That All Network Components Are Redundant
- Use Load Balancers to Distribute Incoming Requests

Ensure That All Network Components Are Redundant

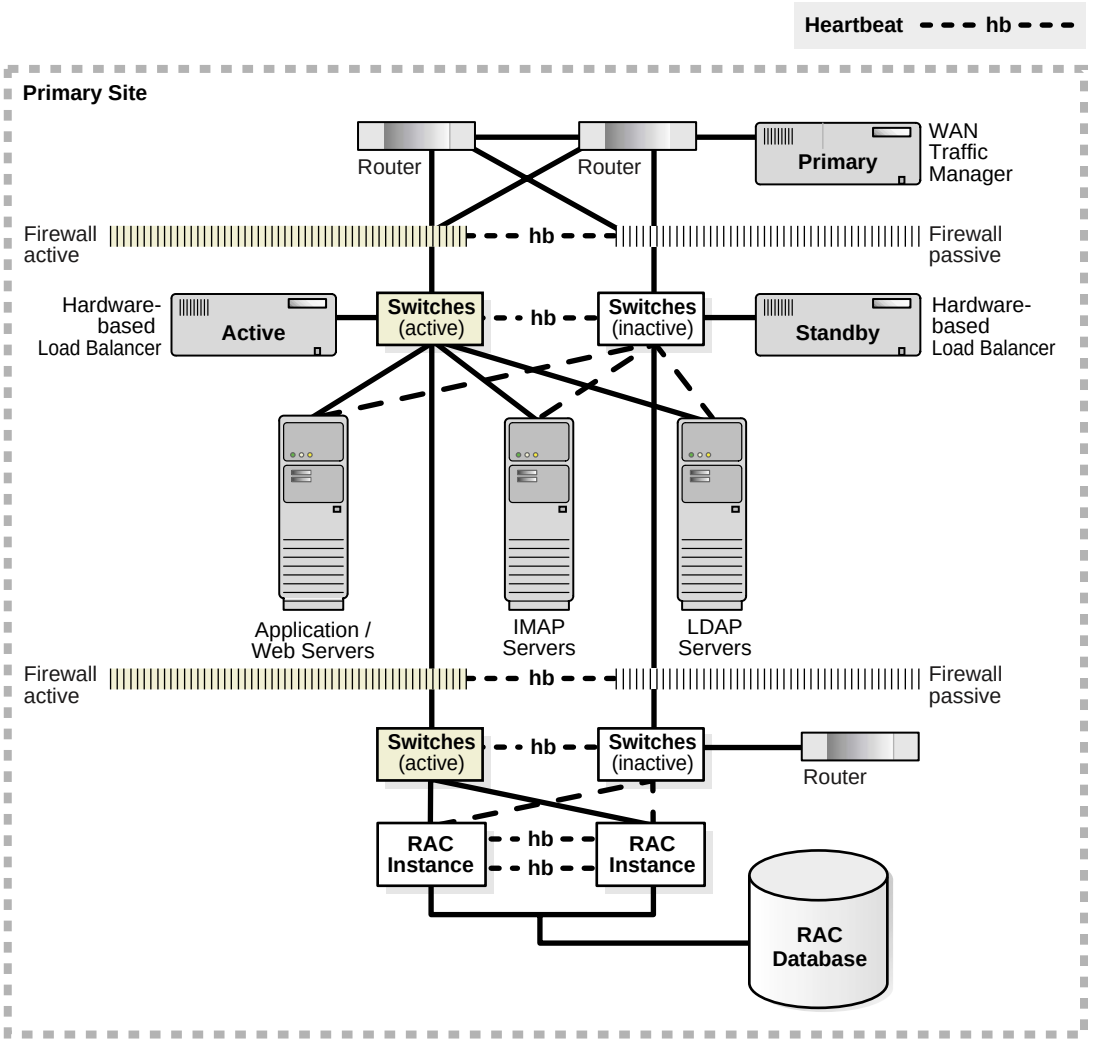
Table 6–2 describes the necessary redundant network components that are illustrated in Figure 6–1.

Table 6–2 Redundant Network Components

Component	Fault Tolerance
Exterior connectivity (including remote IT staff)	Multiple internet service providers (ISP) ensure that an ISP failure does not make your site unavailable. Note: The cables should not be housed within the same trunk into your facility. Otherwise, a saw or shovel can cut connections to multiple ISPs simultaneously.
WAN traffic manager	Use a primary and a backup WAN traffic manager. (Not used with "Database only" or "RAC only" environments) Environments using Data Guard contain a primary and a backup WAN traffic manager at each site.
Application load balancer	Implement redundant application load balancers for directing client requests to the application server tier. The secondary load balancer should be identically configured with a heartbeat to the primary load balancer. If the primary load balancer fails, then the secondary load balancer initiates a takeover function.
Firewall	Implement redundant firewalls and load-balance across the pair. Because the connections through a firewall tend to be longer (socket connections versus HTTP), the load-balanceing decision needs to be made when the session is initiated and maintained during the session. Check with your firewall vendor for available functionality.
Middle-tier application server	Implement application server farms. This provides fault tolerance and scalability.

Figure 6–1 depicts a single site in an MAA environment, emphasizing the redundant network components.

Figure 6–1 Network Configuration



Use Load Balancers to Distribute Incoming Requests

Application layer load balancers sit logically in front of the application server farm and publish to the outside world a single IP address for the service running on a group of application servers. All requests are initially handled by the load balancer, which then distributes them to a server within the application server farm. End

users only need to address their requests to the published IP address; the load balancer determines which server should handle the request.

If one of the middle-tier application servers cannot process a request, the load balancers route all subsequent requests to functioning servers. The load balancers should be redundant to avoid being a single point of failure because all client requests pass through a single hardware-based load balancer. Because failure of that piece of hardware is detrimental to the entire site, a backup load balancer is configured that has a heartbeat with the primary load balancer. With two load balancers, one is configured as standby and becomes active only if the primary load balancer becomes unavailable.

Network Configuration Best Practices for RAC

This recommendation applies to RAC environments.

Classify Network Interfaces Using the Oracle Interface Configuration Tool

Use the Oracle Interface Configuration (OIFCFG) tool to classify network interfaces as public, cluster interconnect, or storage so that RAC properly selects a network interface for internode network traffic.

See Also: *Oracle Real Application Clusters Administrator's Guide*

Network Configuration Best Practices for Data Guard

The following recommendations apply to Data Guard:

- Configure System TCP Parameters Appropriately
- Use WAN Traffic Managers to Provide Site Failover Capabilities

Configure System TCP Parameters Appropriately

Configure system TCP parameters that control the sending and receiving buffer sizes so that the bandwidth between sites can be fully utilized for log transport services. The proper buffer size is often governed by the bandwidth delay product (BDP) formula, particularly when using a high-speed, high-latency network.

See Also:

- "Configuration Best Practices for Data Guard" on page 7-13
- Your operating system documentation for details about changing TCP parameters that contorl the sending and receiving buffer sizes

Use WAN Traffic Managers to Provide Site Failover Capabilities

WAN traffic managers provide the initial access to the services located at the primary site. These managers are implemented at the primary and secondary sites to provide site failover capabilities when the primary site becomes completely unavailable. Geographically separating the WAN traffic managers on separate networks reduces the impact of network failures or a natural disaster that causes the primary site to become unavailable.

See Also: "Complete or Partial Site Failover" on page 10-3

7

Oracle Configuration Best Practices

This chapter describes Oracle configuration best practices. It includes the following sections:

- Configuration Best Practices for the Database
- Configuration Best Practices for Real Application Clusters
- Configuration Best Practices for Data Guard
- Configuration Best Practices for MAA
- Recommendations for Backup and Recovery
- Recommendations for Fast Application Failover

> **See Also:** Appendix B, "Database SPFILE and Oracle Net Configuration File Samples" for a complete example of database parameter settings

Configuration Best Practices for the Database

The practices that are recommended in this section affect the performance, availability, and MTTR of your system. These practices apply to the single-instance database, RAC-only, Data Guard-only, and Maximum Availability architectures described in Chapter 4, "High Availability Architectures". The recommendations in this section are identical for the primary and standby databases when Oracle Data Guard is used. Some of these practices may reduce performance, but they are necessary to reduce or avoid outages. The minimal performance impact is outweighed by the reduced risk of corruption or the performance improvement for recovery.

This section includes the following recommendations:

Use Two Control Files
Set CONTROL_FILE_RECORD_KEEP_TIME Large Enough
Configure the Size of Redo Log Files and Groups Appropriately
Multiplex Online Redo Log Files
Enable ARCHIVELOG Mode
Enable Block Checksums
Enable Database Block Checking
Log Checkpoints to the Alert Log
Use Fast-Start Checkpointing to Control Instance Recovery Time
Capture Performance Statistics About Timing
Use Automatic Undo Management
Use Locally Managed Tablespaces
Use Automatic Segment Space Management
Use Temporary Tablespaces and Specify a Default Temporary Tablespace
Use Resumable Space Allocation
Use a Flash Recovery Area
Enable Flashback Database
Set Up and Follow Security Best Practices
Use the Database Resource Manager
Use a Server Parameter File

Use Two Control Files

Maintain two copies of the control file. If a single control file is damaged, then any Oracle instance fails when it attempts to access the damaged or missing control file. If another copy of the current control file is available, then an instance can be easily restarted after copying the good control file to the location of the bad control file. Database recovery is unnecessary.

See Also: *Oracle Database Administrator's Guide*

Set CONTROL_FILE_RECORD_KEEP_TIME Large Enough

Set the `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter to a value that enables all on-disk backup information to be retained in the control file. Allocate 200 MB for each control file. `CONTROL_FILE_RECORD_KEEP_TIME` specifies the number of days that records are kept within the control file before becoming a candidate for reuse. Set the `CONTROL_FILE_RECORD_KEEP_TIME` value to slightly longer than the oldest backup file that you intend to keep on disk, as determined by the size of the flash recovery area. For example, if the flash recovery area is sized to maintain two full backups that are taken every 7 days, as well as daily incremental backups and archived redo log files, then set `CONTROL_FILE_RECORD_KEEP_TIME` to a value like 21 or 30. Records older than this will be reused. However, the backup metadata will still be available in the RMAN recovery catalog.

See Also: *Oracle Database Backup and Recovery Advanced User's Guide*

Configure the Size of Redo Log Files and Groups Appropriately

All online redo log files should be the same size and configured to switch approximately once an hour during normal activity. They should switch no more frequently than every 20 minutes during peak activity.

There should be a minimum of four online log groups to prevent `LGWR` from waiting for a group to be available following a log switch. A group may be unavailable because a checkpoint has not yet completed or the group has not yet been archived.

See Also:

- *Oracle Database Administrator's Guide*
- *Oracle Data Guard Concepts and Administration*
- "Configuration Best Practices for Data Guard" on page 7-13

Multiplex Online Redo Log Files

Use Oracle log multiplexing to create multiple redo log members in each redo group. This protects against a failure involving the redo log, such as a disk corruption that exists on both sides of the disk mirror for one of the members, or a

user error that accidentally removes a member. If at least one redo log member is available, then the instance can continue to function.

See Also: *Oracle Database Administrator's Guide*

Enable ARCHIVELOG Mode

ARCHIVELOG mode enables the database to be backed up while it is online and is necessary to recover the database to a point in time later than what has already been restored.

Architectures that include Oracle Data Guard require that the production database run in ARCHIVELOG mode before a standby database is instantiated. ARCHIVELOG mode is required to maintain a standby database.

See Also: *Oracle Database Administrator's Guide* for more information about using automatic archiving

Enable Block Checksums

By default, Oracle always tests the data blocks that it reads from disk. Enabling data and log block checksums by setting DB_BLOCK_CHECKSUM to TRUE enables Oracle to detect other types of corruption caused by underlying disks, storage systems, or I/O systems. Before a data block is written to disk, a checksum is computed and stored in the block. When the block is subsequently read from disk, the checksum is computed again and compared with the stored checksum. Any difference is treated as a media error and an ORA-1578 error is signaled. Block checksums are always maintained for the SYSTEM tablespace.

In addition to enabling data block checksums, Oracle also calculates a checksum for every redo log block before writing it to the current log. Redo record corruptions are found as soon as the log is archived. Without this option, a corruption in a redo log can go unnoticed until the log is applied to a standby database or until a backup is restored and rolled forward through the log containing the corrupt log block.

RMAN also calculates checksums when taking backups to ensure that all blocks being backed up are validated.

Turning on this feature typically has minimal overhead. Measure the performance impact with your workload on a test system and ensure that the performance impact is acceptable before introducing this feature on an active database.

See Also: *Oracle Database Administrator's Guide*

Enable Database Block Checking

Enable database block checking by setting DB_BLOCK_CHECKING to TRUE. When block checking is enabled, whenever a block is modified, Oracle verifies that the block is self-consistent. If it is inconsistent, then the block is marked corrupt, an ORA-1578 error is returned, and a trace file is created containing details of the problem. Without block checking enabled, corruptions can go undetected until the block is accessed again. Block checking for the SYSTEM tablespace is always enabled.

Block checking can often prevent memory and data corruption. Turning on this feature typically causes an additional 1 percent to 10 percent overhead, depending on workload. Measure the performance impact on a test system using your workload and ensure that it is acceptable before introducing this feature on an active database.

To ensure that blocks are not corrupted externally to Oracle, use one of the following:

- RMAN BACKUP command with the VALIDATE option
- DBVERIFY utility
- ANALYZE TABLE *tablename* VALIDATE STRUCTURE CASCADE SQL statement

See Also: *Oracle Database Administrator's Guide*

Log Checkpoints to the Alert Log

Checkpoint activity should be logged to the alert log by setting LOG_CHECKPOINT_TO_ALERT to TRUE. Monitor checkpoint activity to ensure that a current checkpoint completes before the next checkpoint starts.

See Also: *Oracle Database Reference*

Use Fast-Start Checkpointing to Control Instance Recovery Time

Fast-start checkpointing refers to the periodic writes by the database writer (DBW*n*) processes. DBW*n* processes write changed data blocks from the Oracle buffer cache to disk and advance the thread checkpoint. With fast-start checkpointing, the checkpoint continually advances so that recovery time from instance or node failure occurs predictably.

Oracle Database 10*g* supports automatic checkpoint tuning which takes advantage of periods of low I/O usage to advance checkpoints and therefore improve availability. Automatic checkpoint tuning is in effect if the FAST_START_MTTR_TARGET database initialization parameter is unset. Observe the following recommendations to take advantage of automatic checkpoint tuning:

- If it is necessary to control the time to recover from an instance or node failure, then set FAST_START_MTTR_TARGET to the desired MTTR in seconds.
- If targeting a specific MTTR is unnecessary, then leave FAST_START_MTTR_TARGET unset to enable automatic checkpoint tuning.
- Fast-start checkpointing can be disabled by setting FAST_START_MTTR_TARGET=0. Disable fast-start checkpointing only when system I/O capacity is insufficient with fast-start checkpointing enabled and achieving a target MTTR is not important.

Enabling fast-start checkpointing increases the average number of writes per transaction that DBW*n* issues for a given workload (when compared with disabling fast-start checkpointing). However, if the system is not already near or at its maximum I/O capacity, then fast-start checkpointing has a negligible impact on performance. The percentage of additional DBW*n* writes with very aggressive fast-start checkpointing depends on many factors, including the workload, I/O speed and capacity, CPU speed and capacity, and the performance of previous recoveries.

Monitor the V$MTTR_TARGET_ADVICE view for advisory information and an estimate of the number of additional I/O operations that would occur under different FAST_START_MTTR_TARGET values. You should also test various FAST_START_MTTR_TARGET settings (such as 0, 1, 90, 180, 270, 3600, and unset) under load to determine the runtime impact (for example, the amount of increased DBW*n* write activity) on a particular system and the instance recovery time achieved with that setting.

If FAST_START_MTTR_TARGET is set to a low value, then fast-start checkpointing is more aggressive, and the average number of writes per transaction that DBW*n* issues is higher in order to keep the thread checkpoint sufficiently advanced to meet the requested MTTR. Conversely, if FAST_START_MTTR_TARGET is set to a high value, or if automatic checkpoint tuning is in effect (that is, FAST_START_MTTR_TARGET is unset), then fast-start checkpointing in less aggressive, and the average number of writes per transaction that DBW*n* issues is lower.

Fast-start checkpointing can be explicitly disabled by setting FAST_START_MTTR_TARGET=0. Disabling fast-start checkpointing leads to the fewest average number

of writes per transaction for DBW*n* for a specific workload and configuration, but also results in the highest MTTR.

When you enable fast-start checkpointing, remove or disable (set to 0) the following initialization parameters: LOG_CHECKPOINT_INTERVAL, LOG_CHECKPOINT_TIMEOUT, FAST_START_IO_TARGET.

See Also: *Oracle Database Performance Tuning Guide*

Capture Performance Statistics About Timing

Set the TIMED_STATISTICS initialization parameter to TRUE to capture Oracle event timing data. This parameter is set to TRUE by default if the STATISTICS_LEVEL database parameter is set to its default value of TYPICAL. Effective data collection and analysis is essential for identifying and correcting system performance problems. Oracle provides several tools that allow a performance engineer to gather information about instance and database performance. Setting TIMED_STATISTICS to TRUE is essential to effectively using the Oracle tools.

See Also:

- *Oracle Database Performance Tuning Guide* for information about Oracle's performance methodology and new performance tuning features
- *Oracle Enterprise Manager Concepts* for information about monitoring and diagnostic tools available with Oracle Enterprise Manager
- *PL/SQL Packages and Types Reference* for information about the DBMS_ADVISOR, DBMS_SQLTUNE, and DBMS_WORKLOAD_REPOSITORY PL/SQL packages

Use Automatic Undo Management

With automatic undo management, the Oracle server effectively and efficiently manages undo space, leading to lower administrative complexity and cost. When Oracle internally manages undo segments, undo block and consistent read contention are eliminated because the size and number of undo segments are automatically adjusted to meet the current workload requirement.

To use automatic undo management, set the following parameters:

- UNDO_MANAGEMENT = AUTO

- UNDO_RETENTION is the desired time in seconds to retain undo data. It must be the same on all instances.
- UNDO_TABLESPACE should specify a unique undo tablespace for each instance.

Advanced object recovery features, such as Flashback Query, Flashback Version Query, Flashback Transaction Query, and Flashback Table, require automatic undo management. These features depend on the UNDO_RETENTION setting. Retention is specified in units of seconds. By default, Oracle automatically tunes undo retention by collecting database use statistics and estimating undo capacity needs. You can affect this automatic tuning by setting the UNDO_RETENTION initialization parameter. The default value of UNDO_RETENTION is 900. You do not need to set this parameter if you want Oracle to tune undo retention. The UNDO_RETENTION value can be changed dynamically at any time by using the ALTER SYSTEM statement.

Setting UNDO_RETENTION does not guarantee that undo data will be retained for the specified period of time. If undo data is needed for transactions, then the UNDO_RETENTION period is reduced so that transactions can receive the necessary undo data.

You can guarantee that unexpired undo data is not overwritten even if it means that future operations that need to generate undo data will fail. This is done by specifying the RETENTION GUARANTEE clause for the undo tablespace when it is created by either the CREATE DATABASE or CREATE UNDO TABLESPACE statement. Alternatively, you can later specify this clause in an ALTER TABLESPACE statement.

With the retention guarantee option, the undo guarantee is preserved even if there is need for DML activity. (DDL statements are still allowed.) If the tablespace is configured with less space than the transaction throughput requires, the following four things will occur in this sequence:

1. If you have an autoextensible file, then it will automatically grow to accommodate the retained undo data.
2. A warning alert is issued at 85 percent full.
3. A critical alert is issued at 97 percent full.
4. Transactions receive an out-of-space error.

> **See Also:** *Oracle Database Administrator's Guide* for more information about the UNDO_RETENTION setting and the size of the undo tablespace

Use Locally Managed Tablespaces

Locally managed tablespaces perform better than dictionary-managed tablespaces, are easier to manage, and eliminate space fragmentation concerns. Locally managed tablespaces use bitmaps stored in the datafile headers and, unlike dictionary managed tablespaces, do not contend for centrally managed resources for space allocations and de-allocations.

See Also: *Oracle Database Administrator's Guide*

Use Automatic Segment Space Management

Automatic segment space management simplifies space administration tasks, thus reducing the chance of human error. An added benefit is the elimination of performance tuning related to space management. It facilitates management of free space within objects such as tables or indexes, improves space utilization, and provides significantly better performance and scalability with simplified administration. The automatic segment space management feature is available only with permanent locally managed tablespaces.

See Also: *Oracle Database Administrator's Guide*

Use Temporary Tablespaces and Specify a Default Temporary Tablespace

Temporary tablespaces improve the concurrency of multiple sort operations, reduce sort operation overhead, and avoid data dictionary space management operations altogether. This is a more efficient way of handling temporary segments, from the perspective of both system resource usage and database performance.

A default temporary tablespace should be specified for the entire database to prevent accidental exclusion of the temporary tablespace clause. This can be done at database creation time by using the `DEFAULT TEMPORARY TABLESPACE` clause of the `CREATE DATABASE` statement or after database creation by the `ALTER DATABASE` statement. Using the default temporary tablespace ensures that all disk sorting occurs in a temporary tablespace and that other tablespaces are not mistakenly used for sorting.

See Also: *Oracle Database Administrator's Guide*

Use Resumable Space Allocation

Resumable space allocation provides a way to suspend and later resume database operations if there are space allocation failures. The affected operation is suspended

instead of the database returning an error. No processes need to be restarted. When the space problem is resolved, the suspended operation is automatically resumed.

Set the RESUMABLE_TIMEOUT initialization parameter to the number of seconds of the retry time.

See Also: *Oracle Database Administrator's Guide*

Use a Flash Recovery Area

The flash recovery area is an Oracle-managed directory, file system, or automatic storage management disk group that provides a centralized disk location for backup and recovery files. The flash recovery area is defined by setting the following database initialization parameters:

- DB_RECOVERY_FILE_DEST: Default location for the flash recovery area
- DB_RECOVERY_FILE_DEST_SIZE: Specifies (in bytes) the hard limit on the total space to be used by target database recovery files created in the recovery area location

The bigger the flash recovery area, the more beneficial it becomes. The minimum recommended disk limit is the sum of the database size, the size of incremental backups, the size of all archived redo logs that have not been copied to tape, and the size of the flashback logs.

See Also: *Oracle Database Backup and Recovery Advanced User's Guide* and *Oracle Database Backup and Recovery Basics* for detailed information about sizing the flash recovery area and setting the retention period

Enable Flashback Database

Flashback Database is a revolutionary recovery feature that operates on only the changed data, thereby making the time to correct an error equal to the time to cause the error without recovery time being a function of the database size. You can flash back a database from both RMAN and SQL*Plus with a single command instead of a complex procedure. Flashback Database is similar to conventional point-in-time recovery in its effects, enabling you to return a database to its state at a time in the recent past. However, Flashback Database is much faster than point-in-time recovery, because it does not require restoring datafiles from backup or extensive application of redo data.

To enable Flashback Database, set up a flash recovery area, and set a flashback retention target (DB_FLASHBACK_RETENTION_TARGET initialization parameter), to specify how far back into the past in minutes you want to be able to restore your database using Flashback Database. To enable Flashback Database, execute the ALTER DATABASE FLASHBACK ON statement. It is important to note that the flashback retention target is a target, not an absolute guarantee that Flashback Database will be available. If your flash recovery area is not large enough to hold required files such as archived redo logs and other backups, then flashback logs may be deleted to make room in the flash recovery area for these required files. To determine how far you can flash back at any one time, query the V$FLASHBACK_DATABASE_LOG view. If you have a standby database, then set FLASHBACK_RETENTION_TIME to be the same for both primary and standby databases.

> **See Also:** *Oracle Database Backup and Recovery Advanced User's Guide*

Set Up and Follow Security Best Practices

The biggest threat to corporate data comes from employees and contractors with internal access to networks and facilities. Corporate data is one of a company's most valuable assets that can be at grave risk if placed on a system or database that does not have proper security measures in place. A well-defined security policy can help protect your systems from unwanted access and protect sensitive corporate information from sabotage. Proper data protection reduces the chance of outages due to security breeches.

In addition to the "High Availability" section in Chapter 9, "Oracle Security Products and Features", the *Oracle Security Overview* manual is a high-level guide to technical security solutions for the data security challenge. Consult Part II, "Technical Solutions to Security Risks" of the *Oracle Security Overview* for an overview of techniques for implementing security best practices. For a much more detailed view of security policies, checklists, guidelines, and features, see the *Oracle Database Security Guide*

> **See Also:** *Oracle Security Overview* and *Oracle Database Security Guide*

Use the Database Resource Manager

The Database Resource Manager gives database administrators more control over resource management decisions, so that resource allocation can be aligned with the business objectives of an enterprise. The Database Resource Manager provides the

ability to prioritize work within the Oracle system. Availability of the database encompasses both its functionality and performance. If the database is available but users are not getting the level of performance they need, then availability and service level objectives are not being met. Application performance, to a large extent, is affected by how resources are distributed among the applications that access the database. The main goal of the Database Resource Manager is to give the Oracle database server more control over resource management decisions, thus circumventing problems resulting from inefficient operating system management and operating system resource managers.

See Also: *Oracle Database Administrator's Guide*

Use a Server Parameter File

The server parameter file (SPFILE) enables a single, central parameter file to hold all of the database initialization parameters associated with all of the instances associated with a database. This provides a simple, persistent, and robust environment for managing database parameters.

An SPFILE is required when using the Data Guard Broker.

See Also:

- *Oracle Database Administrator's Guide*
- *Oracle Real Application Clusters Administrator's Guide*
- *Oracle Data Guard Broker*
- Appendix B, "Database SPFILE and Oracle Net Configuration File Samples"

Configuration Best Practices for Real Application Clusters

The practices that are recommended in this section affect the performance, availability, and MTTR of your system. These practices build on the single instance database configuration best practices. The practices are identical for the primary and standby databases if they are used with Data Guard in the MAA architecture. Some of these practices may reduce performance levels, but they are necessary to reduce or avoid outages. The minimal performance impact is outweighed by the reduced risk of corruption or the performance improvement for recovery.

The rest of this section includes the following topics:

- Register All Instances with Remote Listeners

- Do Not Set CLUSTER_INTERCONNECTS Unless Required for Scalability

See Also: *Oracle Real Application Clusters Administrator's Guide*

Register All Instances with Remote Listeners

The listeners should be cross-registered by using the `REMOTE_LISTENER` parameter so that all listeners know about all services and in which instances the services are running. The listeners should use server-side load balancing, which can be based on session count for connection. The listeners must be listening on the virtual IP addresses and on the cluster alias, when it is available. The listeners must *not* listen on the hostname. Listening on the hostname will result in disconnected sessions when virtual IPs are relocated automatically back to their owning nodes.

See Also:

- *Oracle Real Application Clusters Administrator's Guide*
- *Oracle Net Services Administrator's Guide*

Do Not Set CLUSTER_INTERCONNECTS Unless Required for Scalability

The `CLUSTER_INTERCONNECTS` initialization parameter should be set only if there is more than one cluster interconnect and the default cluster interconnect does not meet the throughput requirements of the RAC database. When `CLUSTER_INTERCONECTS` is set to more than one network address, Oracle load-balances across the interfaces. However, there are no automatic failover capabilities employed by Oracle, requiring that all interfaces be available for a properly functioning database environment.

See Also: *Oracle Real Application Clusters Administrator's Guide*

Configuration Best Practices for Data Guard

These practices build on the recommendations for configuring the single-instance database. The proper configuration of Oracle Data Guard Redo Apply and SQL Apply is essential to ensuring that all standby databases work properly and perform their roles within service levels after switchovers and failovers. Most Data Guard configuration settings can be made through the Oracle Enterprise Manager. For more advanced, less frequently used Data Guard configuration parameters, the Data Guard Broker command-line interface or SQL*Plus can be used.

Data Guard enables you to use either a physical standby database or a logical standby database or both, depending on the business requirements. A physical standby database provides a physically identical copy of the primary database, with on-disk database structures that are identical to the primary database on a block-for-block basis. The database schema, including indexes, are the same. A physical standby database is kept synchronized with the primary database by applying the redo data received from the primary database.

A logical standby database contains the same logical information as the production database, although the physical organization and structure of the data can be different. It is kept synchronized with the primary database by transforming the data in the redo log files received from the primary database into SQL statements and then executing the SQL statements on the standby database. A logical standby database can be used for other business purposes in addition to disaster recovery requirements.

Table 7–1 can help you determine which type of standby database to use.

Table 7–1 Determining the Standby Database Type

Questions	Recommendations
1. Do you have any datatypes that are not supported by the logical standby database?	Run the following query: `SELECT DISTINCT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED ORDER BY OWNER, TABLE_NAME;` Rows returned - Use a physical standby database or investigate changing to supported datatypes No rows returned - Go to next question
2. Do you need to have the standby database open for read-only or read / write access? **See Also**: "Oracle9*i* Data Guard: SQL Apply Best Practices" at `http://otn.oracle.com/deploy/availability/htdocs/maa.htm`	Yes - Evaluate a logical standby database No - Use a physical standby database

Table 7–2 shows the recommendations for configuration that apply to both logical and physical standby databases and those that apply to only logical and only physical.

Table 7–2 Recommendations for Configuring Standby Databases

Recommendations for Both Physical and Logical Standby Databases	Recommendations for Physical Standby Databases Only	Recommendations for Logical Standby Databases Only
Use a Simple, Robust Archiving Strategy and Configuration	Tune Media Recovery Performance	Use Supplemental Logging and Primary Key Constraints
Use Multiplexed Standby Redo Logs and Configure Size Appropriately	-	Set the MAX_SERVERS Initialization Parameter
Enable FORCE LOGGING Mode	-	Increase the PARALLEL_MAX_ SERVERS Initialization Parameter
Use Real Time Apply	-	Set the TRANSACTION_ CONSISTENCY Initialization Parameter
Configure the Database and Listener for Dynamic Service Registration	-	Skip SQL Apply for Unnecessary Objects
Tune the Network in a WAN Environment	-	-
Determine the Data Protection Mode	-	-
Conduct a Performance Assessment with the Proposed Network Configuration	-	-
Use a LAN or MAN for Maximum Availability or Maximum Protection Modes	-	-
Set SYNC=NOPARALLEL / PARALLEL Appropriately	-	-
Use ARCH for the Greatest Performance Throughput	-	-
Use the ASYNC Attribute with a 50 MB Buffer for Maximum Performance Mode	-	-
Evaluate SSH Port Forwarding with Compression	-	-

Table 7–2 Recommendations for Configuring Standby Databases (Cont.)

Recommendations for Both Physical and Logical Standby Databases	Recommendations for Physical Standby Databases Only	Recommendations for Logical Standby Databases Only
Set LOG_ARCHIVE_LOCAL_ FIRST to TRUE	-	-
Provide Secure Transmission of Redo Data	-	-
Set DB_UNIQUE_NAME	-	-
Set LOG_ARCHIVE_CONFIG Correctly	-	-

See Also:

- *Oracle Data Guard Concepts and Administration*
- *Oracle Database Reference*

Use a Simple, Robust Archiving Strategy and Configuration

This archiving strategy is based on the following assumptions:

- Every instance uses the flash recovery area.
- The production instances archive remotely to only one apply instance.

Table 7–3 describes the recommendations for a robust archiving strategy.

Table 7–3 Archiving Recommendations

Recommendation	Description

Archiving must be started on the primary database	Maintaining a standby database requires archiving to be enabled and started on the primary database.
	`SQL> SHUTDOWN IMMEDIATE`
	`SQL> STARTUP MOUNT;`
	`SQL> ALTER DATABASE ARCHIVELOG;`
	`SQL> ALTER DATABASE OPEN;`

Remote archiving must be enabled.	`REMOTE_ARCHIVE_ENABLE=TRUE`

Table 7–3 Archiving Recommendations (Cont.)

Recommendation	Description
Use a consistent log format (LOG_ARCHIVE_FORMAT).	LOG_ARCHIVE_FORMAT should have the thread, sequence, and resetlogs ID attributes and should be consistent across all instances. %S instructs the format to fill the prefix of the sequence number with leading zeros. If the flash recovery is used, then this format is ignored. For example: LOG_ARCHIVE_FORMAT=arch_%t_%S_%r.arc
Local archiving is done first by the archiver process (ARCH).	Using ARCH reduces the work for LGWR. The default setting for LOG_ARCHIVE_LOCAL_FIRST is TRUE, which means that after the redo log has been completely and successfully archived to at least one local destination, it will then be transmitted to the remote destination. Using the flash recovery area implies that LOG_ARCHIVE_DEST_10 is used for local archiving.
Remote archiving should be done to only one standby instance and node for each standby RAC database.	All production instances archive to one standby destination, using the same net service name. Oracle Net Services connect-time failover is used if you want to automatically switch to the secondary standby host when the primary standby instance has an outage.
The standby archive destination should use the flash recovery area.	For simplicity, the standby archive destination (STANDBY_ARCHIVE_DEST) should use the flash recovery area, which is the same as the directory for the local archiving. Because SRLs are present, the standby ARCH process writes to the local archive destination.
The logical standby archive destination cannot use the flash recovery area.	For a logical standby database, STANDBY_ARCHIVE_DEST cannot use the flash recovery area. Set STANDBY_ARCHIVE_DEST to an explicit archive directory.
Specify role-based destinations with the VALID_FOR attribute	The VALID_FOR attribute enables you to configure destination attributes for both the primary and the standby database roles in one server parameter file (SPFILE), so that the Data Guard configuration operates properly after a role transition. This simplifies switchovers and failovers by removing the need to enable and disable the role-specific parameter files after a role transition. **See Also:** Appendix B, "Database SPFILE and Oracle Net Configuration File Samples"

The following example illustrates the recommended initialization parameters for a primary database communicating to a physical standby database. There are two instances, SALES1 and SALES2, running in maximum protection mode.

```
*.DB_RECOVERY_FILE_DEST=/recoveryarea
```

```
*LOG_ARCHIVE_DEST_1='SERVICE=SALES LGWR SYNC=NOPARALLEL AFFIRM
    REOPEN=15 MAX_FAILURE=10 VALID_FOR=(ONLINE+LOGFILES, ALL ROLES)'
*.LOG_ARCHIVE_DEST_STATE_1='ENABLE'
*.STANDBY_ARCHIVE_DEST='USE_DB_RECOVERY_FILE_DEST'
```

Note the following observations for this example:

- The PARALLEL attribute is used when there are multiple standby destinations. When SYNC is set to PARALLEL, the LGWR process initiates an I/O operation to each standby destination at the same time. Because there is a single standby destination, the NOPARALLEL option is set to reduce overhead.
- The REOPEN=15 MAX_FAILURE=10 setting denotes that if there is a connection failure, then the connection is reopened after 15 seconds and is retried up to 10 times.
- The VALID_FOR clause is used to designate the role for a destination. When the database is in a physical standby role, remote destination LOG_ARCHIVE_DEST_1 is not used because a physical standby database does not use online log files.

The flash recovery area must be accessible to any node within the cluster and use a shared file system technology such as automatic storage management (ASM), a cluster file system, a global file system, or high availability network file system (HA NFS). You can also mount the file system manually to any node within the cluster very quickly. This is necessary for recovery because all archived redo log files must be accessible on all nodes.

On the standby database nodes, recovery from a different node is required when Node 1 fails and cannot be restarted. In that case, any of the existing standby instances residing on a different node can initiate managed recovery. In the worst case, when the standby archived redo log files are inaccessible, the new managed recovery process (MRP) or logical standby process (LSP) on the different node fetches the archived redo log files using the FAL server to retrieve from the production nodes directly.

When configuring hardware vendor shared file system technology, verify the performance and availability implications. Investigate the following issues before adopting this strategy:

- Is the shared file system accessible by any node regardless of the number of node failures?
- What is the performance impact when implementing a shared file system?
- Is there any impact on the interconnect traffic?

Use Multiplexed Standby Redo Logs and Configure Size Appropriately

Standby redo logs (SRLs) should be used on both sites. Use Oracle log multiplexing to create multiple standby redo log members in each standby redo group. This protects against a failure involving the redo log, such as disk corruption that exists on both sides of the disk mirror for one of the members or a user error that accidentally removed a member.

Use this formula to determine the number of SRLs:

```
# of SRLs = sum of all production online log groups per thread + number of threads
```

For example, if a primary database has two instances (threads) and each thread has four online log groups, then there should be ten SRLs. Having one more standby log group for each thread than the number of the online redo log groups for the production database reduces the likelihood that the LGWR for the production instance is blocked because an SRL cannot be allocated on the standby.

The following are additional guidelines for creating SRLs:

- Create the same number of SRLs for both production and standby databases.
- All of the online redo logs and SRLs for both the production and the standby databases should be the same size.
- The SRLs should exist on both production and standby databases.
- In a RAC environment, the SRLs must be on a shared disk.
- In a RAC environment, assign the SRL to a thread when the SRL is created. For example:

```
ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 GROUP 10
   '/dev/vx/rdsk/ha10-dg/DGFUN stbyredo10 01.log' SIZE 50M REUSE;
```

The remote file server (RFS) process for the standby database writes only to an SRL whose size is identical to the size of an online redo log for the production database. If it cannot find an appropriately sized SRL, then RFS creates an archived redo log file directly instead and logs the following message in the alert log:

```
No standby redo log files of size <#> blocks available.
```

Enable FORCE LOGGING Mode

When the production database is in FORCE LOGGING mode, all database changes are logged except for those in temporary tablespaces and temporary segments. FORCE LOGGING mode ensures that the standby database remains consistent with

the production database. If this is not possible because you require the load performance with NOLOGGING operations, then you must ensure that the corresponding standby datafiles are subsequently synchronized. After completing the nologging operations, a production backup of the affected datafiles needs to replace the corresponding standby datafiles. Before the file transfer, the physical standby database must stop recovery and the logical standby database must temporarily take the affected tablespaces offline.

You can enable force logging immediately by issuing an ALTER DATABASE FORCE LOGGING statement. If you specify FORCE LOGGING, then Oracle waits for all ongoing unlogged operations to finish.

See Also:

- *Oracle Database Administrator's Guide*
- *Oracle Data Guard Concepts and Administration*

Use Real Time Apply

Using real time apply enables the log apply services to apply redo data (physical standby database) or SQL (logical standby database) as it is received without waiting for the current standby redo log file to be archived. This results in faster switchover and failover times because the standby redo log files are applied to the standby database before failover or switchover begins.

For a physical standby database, use the following SQL statement

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE;
```

For a logical standby database, use the following SQL statement:

```
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

See Also: *Oracle Data Guard Concepts and Administration*

Configure the Database and Listener for Dynamic Service Registration

The setting for the SERVICE attribute of the LOG_ARCHIVE_DEST_2 initialization parameter and the settings for the FAL_SERVER and FAL_CLIENT initialization parameters depend on a proper Oracle Net configuration. For the Oracle Data Guard transport service and the gap resolution feature to work, the SPFILE, listener.ora, tnsnames.ora, and sqlnet.ora files must be consistent.

See Also:

- Appendix B, "Database SPFILE and Oracle Net Configuration File Samples"
- MetaLink Note 76636.1 at `http://metalink.oracle.com`

The remote archive destination FAL_CLIENT and FAL_SERVER parameters require an Oracle Net service. This service is represented as a net service name entry in the local `tnsnames.ora` file. Notice that the FAL_SERVER and FAL_CLIENT reference the same Oracle network service name. This is possible because the FAL_SERVER service is defined in the standby `tnsnames.ora` file, whereas the FAL_CLIENT service is defined in the primary `tnsnames.ora` file. This works only when you use the Oracle Network Service local naming method. If you are not using the local naming method, then you must have different service names. Furthermore, Oracle recommends using dynamic service registration instead of a static SID list in the listener configuration. To ensure that service registration works properly, the server parameter file should contain the following parameters:

- SERVICE_NAMES for the database service name
- INSTANCE_NAME for the instance name
- LOCAL_LISTENER to specify a nondefault listener address

PMON dynamically registers a database service with the listener. PMON attempts to resolve LOCAL_LISTENER using some naming method. In the case described here, PMON finds the corresponding name in the local `tnsnames.ora` file.

For example:

```
SALES1.INSTANCE_NAME='SALES1'
SALES2.INSTANCE_NAME='SALES2'
*.LOG_ARCHIVE_DEST_2='SERVICE=SALES LGWR SYNC=NOPARALLEL AFFIRM REOPEN=15 MAX_FAILURE=10'
*.LOCAL_LISTENER='SALES_lsnr'
*.SERVICE_NAMES='SALES'    # required for service registration
*.FAL_SERVER='SALES'
*.FAL_CLIENT='SALES'
```

The `listener.ora` file should be identical for each primary and secondary host except for HOST settings. Because service registration is used, there is no need for statically configured information.

The local `tnsnames.ora` file should contain the net service names and the local listener name translation. To use the same service name on each node, you must use a locally managed `tnsnames.ora` file for the production and standby databases.

On the primary cluster, the `tnsnames.ora` entry, `SERVICE_NAME`, should equal the setting of the `SERVICE_NAMES SPFILE` parameter. If the listener is started after the instance, then service registration does not happen immediately. In this case, issue the `ALTER SYSTEM REGISTER` statement on the database to instruct the `PMON` background process to register the instance with the listeners immediately.

See Also: *Oracle Net Services Administrator's Guide*

Tune the Network in a WAN Environment

Reducing the number of round trips across the network is essential for optimizing the transportation of redo log data to a standby site. With Oracle Net Services it is possible to control data transfer by adjusting the size of the Oracle Net setting for the session data unit (SDU). In a WAN environment, setting the SDU to 32K can improve performance. The SDU parameter designates the size of an Oracle Net buffer before it delivers each buffer to the TCP/IP network layer for transmission across the network. Oracle Net sends the data in the buffer either when requested or when it is full. Oracle internal testing of Oracle Data Guard on a WAN has demonstrated that the maximum setting of 32K (32768) performs best on a WAN. The primary gain in performance when setting the SDU is a result of the reduced number of calls to packet the data.

In addition to setting the `SDU` parameter, network throughput can often be substantially improved by using the `SQLNET.SEND_BUF_SIZE` and `SQLNET.RECV_BUF_SIZE` Oracle Net parameters to increase the size of the network TCP send and receive I/O buffers.

See Also: *Oracle Net Services Administrator's Guide*

Determine the Data Protection Mode

In some situations, a business cannot afford to lose data at any cost. In other situations, the availability of the database may be more important than protecting f data. Some applications require maximum database performance and can tolerate a potential loss of data if a disaster occurs.

Choose one of the following protection modes:

- **Maximum protection mode** guarantees that no data loss will occur if the primary database fails. To ensure that data loss cannot occur, the primary database shuts down if a fault prevents it from writing its redo stream to at least one remote standby redo log.

- **Maximum availability mode** provides the highest level of data protection that is possible without compromising the availability of the primary database.
- **Maximum performance mode** (the default mode) provides the highest level of data protection that is possible without affecting the performance of the primary database. This is accomplished by allowing a transaction to commit as soon as the redo data needed to recover that transaction is written to the local online redo log. The redo data stream of the primary database is also written to at least one standby database, but that redo stream is written asynchronously with respect to the commitment of the transactions that create the redo data. When network links with sufficient bandwidth are used, this mode provides a level of data protection that approaches that of maximum availability mode with minimal impact on primary database performance.

This section includes the following topics:

- Determining the Protection Mode
- Changing the Data Protection Mode

> **See Also:** *Oracle Data Guard Concepts and Administration* for more information about data protection modes

Determining the Protection Mode

To determine the correct data protection mode for your application, ask the questions in Table 7–4.

Table 7–4 Determining the Appropriate Protection Mode

Question	Recommendations
Is data loss acceptable if the primary site fails?	Yes: Use any protection mode. No: Use maximum protection or maximum availability modes.
How much data loss is tolerated if a site is lost?	None: Use maximum protection or maximum availability modes. Some: Use maximum performance mode with ASYNC=*blocks*. The value for the number of blocks determines the maximum amount of possible redo data loss if a site fails.
Is potential data loss between the production and the standby databases tolerated when a standby host or network connection is temporarily unavailable?	Yes: Use maximum performance or maximum availability modes. No: Use maximum protection mode.

Table 7–4 Determining the Appropriate Protection Mode (Cont.)

Question	Recommendations
How far away should the disaster recovery site be from the primary site?	The distance between sites and the network infrastructure between the sites determines network latency. In general, the latency increases with distance. Determine the minimum distance between sites to provide for outage isolation and minimal network latency. Assess what data centers are available for your company, or assess Oracle outsourcing services.
What is the current or proposed network bandwidth and latency between sites?	Bandwidth must be greater than maximum redo generation rate. A guideline for two-way communication is for bandwidth to be 50 percent of the stated line capacity, but you must consider network usage of other applications. Using maximum performance mode with asynchronous log transport or the archiver mitigates the effect on performance.

Changing the Data Protection Mode

The default data protection mode is maximum performance mode. After a failover to the standby database, the protection mode automatically changes to maximum performance mode. Switchover operations do not change the protection mode.

To change the data protection mode from maximum performance to maximum availability or maximum protection, perform the following steps:

1. Change the appropriate initialization parameters. For maximum protection and maximum availability modes, Oracle requires one functional remote or standby archive destination using the `LGWR SYNC` option and a valid net service name during startup. For maximum performance mode, use the `LGWR ASYNC` option with a valid net service name.
2. Shut down the primary database and restart it in mounted mode.
3. Shut down all instances and start a single instance in exclusive mode.

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP MOUNT EXCLUSIVE;
   ```

4. Change the data protection mode explicitly to the desired mode

   ```
   ALTER DATABASE SET STANDBY TO MAXIMIZE [AVAILABILITY | PROTECTION];
   ```

5. Restart all instances.

To change the protection mode from maximum protection to maximum performance or maximum availability, use a statement similar to the following:

```
ALTER DATABASE SET STANDBY TO MAXIMIZE [PERFORMANCE | AVAILABILITY];
```

See Also:

- *Oracle Data Guard Concepts and Administration* for more information about setting the data protection mode for a Data Guard configuration
- *Oracle Data Guard Concepts and Administration* for more information about changing the protection mode in a Data Guard and RAC configuration

Conduct a Performance Assessment with the Proposed Network Configuration

Oracle recommends that you conduct a performance assessment with your proposed network configuration and current (or anticipated) peak redo rate. The network impact between the production and standby databases and the impact on the primary database throughput needs to be understood. Because the network between the production and standby databases is essential for the two databases to remain synchronized, the infrastructure must have the following characteristics:

- Sufficient bandwidth to accommodate the maximum redo generation rate
- Minimal latency to reduce the performance impact on the production database
- Multiple network paths for network redundancy

The required bandwidth of a dedicated network connection is determined by the maximum redo rate of the production database. You also need to account for actual network efficiency. Depending on the data protection mode, there are other recommended practices and performance considerations. Maximum protection mode and maximum availability mode require `LGWR SYNC` transport. Maximum performance protection mode uses the `ASYNC` transport option or the archiver (`ARCH`*n*) instead of `LGWR` to transfer the redo. These recommendations were derived from an Oracle internal performance study that measured the impact of network latency on primary database throughput for each Oracle Data Guard transport option: `ARCH`, `LGWR ASYNC`, and `LGWR SYNC`.

See Also:

`http://otn.oracle.com/deploy/availability/htdocs/maa.htm` for details of the study

The network infrastructure between the primary and secondary sites must be able to accommodate the redo traffic because the production database redo data is updating the physical standby database. If your maximum redo traffic at peak load is 8 MB/second, then your network infrastructure must have sufficient bandwidth

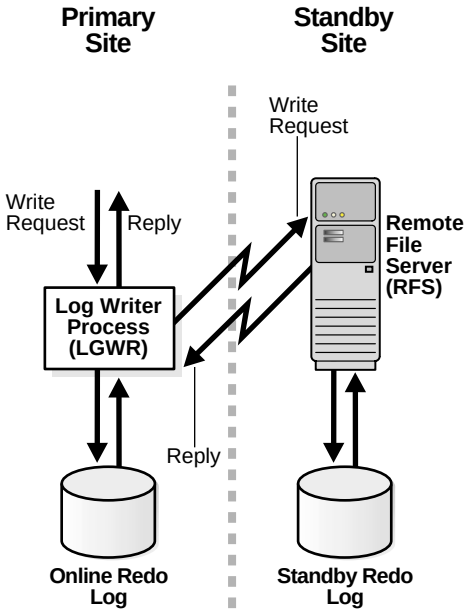
to handle this load. Furthermore, network latency affects overall throughput and response time for OLTP and batch operations.

When you compare maximum protection mode or maximum availability mode with LGWR SYNC operations with maximum performance mode with LGWR ASYNC operations, measure whether performance or throughput will be degraded due to the incurred latency. You should also check whether the new throughput and response time are within your application performance requirements. Distance and the network configuration directly influence latency, while high latency may slow down your potential transaction throughput and increase response time. The network configuration, number of repeaters, the overhead of protocol conversions, and the number of routers also affect the overall network latency and transaction response time.

Use a LAN or MAN for Maximum Availability or Maximum Protection Modes

Maximum availability mode or maximum protection mode require the Oracle Data Guard transport service to use the LGWR SYNC transport option. Network latency is an additional overhead for each LGWR SYNC I/O operation. Figure 7–1 shows that LGWR SYNC writes both locally to the online redo log and remotely through the network to the RFS process to the standby redo logs.

Figure 7–1 LGWR SYNC Operation



The following formulas emphasize that the remote write is always slower than the local write and is the limiting factor when LGWR synchronous writes are occurring.

```
Local write = local write I/O time
Remote write = network round trip time (RTT) + local write I/O time (on standby machine)
```

Using an example in which the network round trip time (RTT) is 20 milliseconds and LGWR synchronous write is configured, every transaction commit time increases by 20 milliseconds. This overhead impacts response time and may affect primary database throughput. Because of the additional overhead incurred by the RTT, a local area network (LAN) or a metropolitan area network (MAN) with an RTT less than or equal to 10 milliseconds should be used for applications that cannot tolerate a change in performance or response time. Whether to use a LAN or MAN depends on the results of the performance assessment.

Set SYNC=NOPARALLEL/PARALLEL Appropriately

With only one remote standby destination within a LAN, with sufficient bandwidth and low latency, Oracle recommends the LGWR SYNC=NOPARALLEL AFFIRM option for the best performance with maximum data protection capabilities. When

no data loss is required and there is only one remote archive destination, SYNC=NOPARALLEL performs better than SYNC=PARALLEL (the default) with a single standby destination.

If SYNC=PARALLEL is used, then the network I/O is initiated asynchronously, so that I/O to multiple destinations can be initiated in parallel. However, after the I/O is initiated, LGWR waits for each I/O operation to complete before continuing. This is, in effect, the same as performing multiple synchronous I/O operations simultaneously. If the Data Guard configuration has more than one standby database, then use a cascading standby architecture to minimize overhead to the primary database. If business requirements stipulate that the production database must transfer to multiple standby databases, then set SYNC=PARALLEL.

See Also:

- *Oracle Data Guard Concepts and Administration* for more information about cascading standby databases
- "Oracle9*i* Data Guard: Primary Site and Network Configuration Best Practices" at `http://otn.oracle.com/deploy/availability/pdf/MAA_DG_NetBestPrac.pdf`. These practices continue to apply to Oracle Database 10*g*.

Use ARCH for the Greatest Performance Throughput

The ARCH attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter provides the greatest performance throughput but the greatest data loss potential. ARCH does not affect primary performance when latency increases as long as the redo logs are configured correctly as described in "Configure the Size of Redo Log Files and Groups Appropriately" on page 7-3. This is recommended for maximum performance data protection mode and is the default.

The effects of latency on primary throughput are detailed in the following white paper.

See Also: "Oracle9*i* Data Guard: Primary Site and Network Configuration Best Practices" at `http://otn.oracle.com/deploy/availability/htdocs/maa.htm` for details about ARCH performance and latency

Use the ASYNC Attribute with a 50 MB Buffer for Maximum Performance Mode

The largest LGWR ASYNC buffer of 50 MB (ASYNC=102400) performs best in a WAN. In LAN tests, different asynchronous buffer sizes did not impact the primary database throughput. Using the maximum buffer size also increases the chance of avoiding timeout messages that result from an "async buffer full" condition in a WAN.

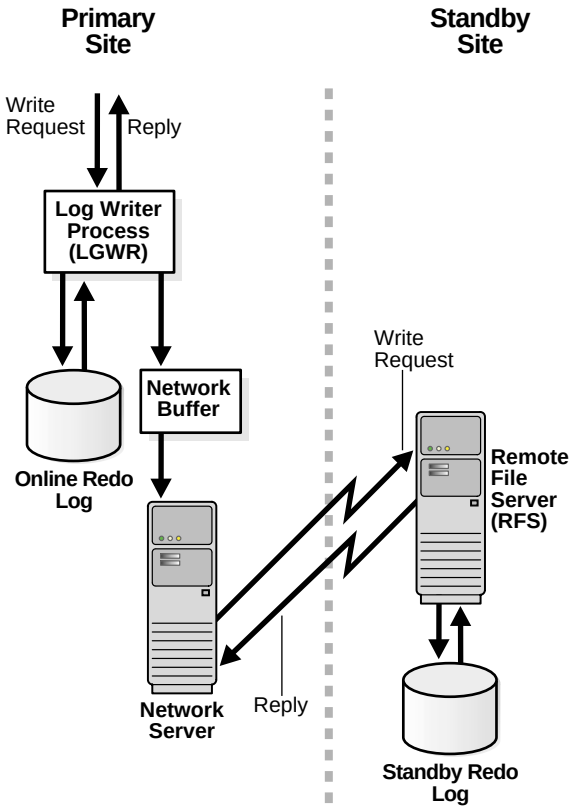
If the network buffer becomes full and remains full for 5 seconds, then the transport times out and converts to the ARCH transport. This condition indicates that the network to the standby destination cannot keep up with the redo generation rate on the primary database. This indicated in the alert log by the following message:

```
'Timing out on NetServer %d prod=%d,cons=%d,threshold=%d"
```

This message indicates that the standby destination configured with the LGWR ASYNC attributes encountered an "async buffer full" condition. When this occurs, log transport services automatically stop using the network server process to transmit the redo data and convert to using the archiver process (ARC*n*) until a log switch occurs. The next log transmission reverts to the ASYNC transport. This change occurs automatically. Using the largest asynchronous network buffer, 30MB, increases the chance of avoiding the transport converting to ARCH and potentially losing more data.

Figure 7–2 shows the architecture when the standby protection mode is set to maximum performance with LGWR ASYNC configuration. The LGWR request is buffered if sufficient space is available in the network buffer. If the production database fails and is inaccessible, then the data in the network buffer is lost, which usually means seconds of data loss in high OLTP applications.

Figure 7–2 LGWR ASYNC Transport Service



Evaluate SSH Port Forwarding with Compression

Evaluate SSH port forwarding with compression for maximum performance mode over a high-latency WAN (RTT greater than 100 milliseconds). Coupled with using `LGWR ASYNC`, the maximum buffer size, SSH with compression reduces the chance of receiving an "async buffer full" timeout. It also reduces network traffic.

See Also:

- "Oracle9*i* Data Guard: Primary Site and Network Configuration Best Practices" at `http://otn.oracle.com/deploy/availability/htdocs/maa.htm`
- MetaLink Note 225633.1 at `http://metalink.oracle.com`

Set LOG_ARCHIVE_LOCAL_FIRST to TRUE

Setting `LOG_ARCHIVE_LOCAL_FIRST` to `TRUE` enables the archiver processes to archive the local online redo log files on the primary database before transmitting the redo data to remote standby destinations. This is especially useful when the network to the standby databases is slow.

This is the default setting for `LOG_ARCHIVE_LOCAL_FIRST`.

Provide Secure Transmission of Redo Data

Because a lack of security can directly affect availability, Data Guard provides a secure environment and prevents tampering with redo data as it is being transferred to the standby database. To enable secure transmission of redo data, set up every database in the Data Guard configuration to use a password file, and set the password for the `SYS` user identically on every system. The following is a summary of steps needed for each database in the Data Guard configuration:

1. Create a password file for each database in the Data Guard configuration.
2. Set the `REMOTE_LOGIN_PASSWORDFILE=[EXCLUSIVE | SHARED]` initialization parameter on each instance.

After you have performed these steps to set up security on every database in the Data Guard configuration, Data Guard transmits redo data only after the appropriate authentication checks using `SYS` credentials are successful. This authentication can be performed even if Oracle Advanced Security is not installed and provides some level of security when shipping redo data. To further protect redo data (for example, to encrypt redo data or to compute an integrity checksum value for redo traffic over the network to disallow redo tampering on the network), Oracle recommends that you install and use Oracle Advanced Security.

See Also:

- *Oracle Data Guard Concepts and Administration* for detailed steps for providing secure transmission of redo data
- *Oracle Advanced Security Administrator's Guide*

Set DB_UNIQUE_NAME

Specify a unique name for the standby database. The name does not change even if the primary and standby databases reverse roles. The DB_UNIQUE_NAME parameter defaults to the value of the DB_NAME parameter.

See Also: *Oracle Data Guard Concepts and Administration*

Set LOG_ARCHIVE_CONFIG Correctly

Specify the DG_CONFIG attribute of the LOG_ARCHIVE_CONFIG initialization parameter so that it lists the DB_UNIQUE_NAME for the primary database and each standby database in the Data Guard configuration. By default, this parameter enables the primary database to send redo data to remote destinations and enables standby databases to receive redo data. The DG_CONFIG attribute must be set to enable the dynamic addition of a standby database to a Data Guard configuration that has a RAC primary database running in either maximum protection or maximum availability mode.

See Also: *Oracle Data Guard Concepts and Administration*

Recommendations for the Physical Standby Database Only

The following recommendation applies only to the physical standby database:

- Tune Media Recovery Performance

Tune Media Recovery Performance

To use Oracle Data Guard with a physical standby database or to use any media recovery operation effectively, you need to tune your database recovery.

See Also: "Oracle9*i* Media Recovery Best Practices" at `http://otn.oracle.com/deploy/availability/htdocs/maa.htm`

Recommendations for the Logical Standby Database Only

The following recommendations apply only to the logical standby database:

- Use Supplemental Logging and Primary Key Constraints
- Set the MAX_SERVERS Initialization Parameter
- Increase the PARALLEL_MAX_SERVERS Initialization Parameter
- Set the TRANSACTION_CONSISTENCY Initialization Parameter
- Skip SQL Apply for Unnecessary Objects

Use Supplemental Logging and Primary Key Constraints

Use supplemental logging and primary key constraints on all production tables.

If your application ensures that the rows in a table are unique, then you can create a disabled primary key RELY constraint on the table. This avoids the overhead of maintaining a primary key on the primary database. To create a disabled RELY constraint on a primary database table, use the ALTER TABLE statement with a RELY DISABLE clause.

To improve the performance of SQL Apply, add an index to the columns that uniquely identify the row on the logical standby database. Failure to do this results in full table scans.

Set the MAX_SERVERS Initialization Parameter

If the logical standby database is being used to remove reporting or decision support operations from the primary database, then you should probably reserve some of the parallel query slaves for such operations. Because the SQL Apply process by default uses all the parallel query slaves, setting the MAX_SERVERS initialization parameter enables a specified number of parallel query slaves to be reserved.

Table 7–5 shows examples of MAX_SERVERS values.

Table 7–5 Examples of MAX_SERVERS Values

PARALLEL_MAX_SERVERS Initialization Parameter	MAX_SERVERS Initialization Parameter	Number of Servers Reserved for Parallel Query Operations	Number of Servers Reserved for SQL Apply Operations
24	Unset	0	24
24	24	0	24
48	24	24	24

It is recommended that MAX_SERVERS be set initially to the larger of the following values: 9 or 3 plus 3 times CPU.

Increase the PARALLEL_MAX_SERVERS Initialization Parameter

Increase the PARALLEL_MAX_SERVERS initialization parameter by the larger of 9 or 3 times CPU on both the primary and standby instances:

```
PARALLEL_MAX_SERVERS=current value + max(9, 3 +(3 x CPU))
```

The PARALLEL_MAX_SERVERS initialization parameter specifies the maximum number of parallel query processes that can be created on the database instance. With the exception of the coordinator process, all the processes that constitute the SQL Apply engine are created from the pool of parallel query processes. The SQL Apply engine, by default, uses all the parallel query processes available on the database instance. This behavior can be overridden by using the logical standby parameters

It is recommended that PARALLEL_MAX_SERVERS be increased by the value of MAX_SERVERS.

Set the TRANSACTION_CONSISTENCY Initialization Parameter

The logical standby database supports the following methods of data application:

- For a reporting or decision support system, use FULL or READ_ONLY transaction consistency.
- For a disaster recovery solution or when the SQL Apply engine needs to catch up, set TRANSACTION_CONSISTENCY to NONE.

If the logical standby database will be used for reporting or decision support operations, then:

- If the standby database has multiple instances (RAC), then choose FULL.

- If the standby database has only one instance (no RAC), then choose READ_ONLY.

Skip SQL Apply for Unnecessary Objects

Database objects that do not need to be replicated to the standby database should be skipped by using the DBMS_LOGSTDBY.SKIP procedure. Skipping such objects reduces the processing of the the SQL Apply engine. Consider this recommendation in a decision support environment.

Configuration Best Practices for MAA

This section recommends configuration practices in addition to the ones that are discussed for the single-instance database, RAC, and Data Guard. These practices are recommended when MAA is employed (RAC and Data Guard are used on both sites).

This section includes the following topics:

- Configure Multiple Standby Instances
- Configure Connect-Time Failover for Network Service Descriptors

Configure Multiple Standby Instances

In an MAA environment, the standby database uses RAC, and multiple standby instances are associated with the same standby database. Having multiple standby instances is not the same as having multiple standby databases. Only one instance can have the managed recovery process (MRP) or the logical standby apply process (LSP). The standby instance with the MRP or LSP is called the primary standby instance. All other standby instances are called secondary standby instances.

Having multiple standby instances for the same database on the cluster provides the following benefits:

- They enable transparent connection failover to a secondary standby instance if connectivity to the primary standby instance fails. In this scenario, the MRP or LSP session is automatically restarted by the Data Guard Broker. If the Broker is not being used, then these processes must be restarted manually on the new primary standby instance.
- They provide a scheduled maintenance solution whenever the primary standby instance and host need to be shut down for maintenance. The secondary

standby can take over and receive logs through Oracle Net service because connect-time failover occurs.

Configure Connect-Time Failover for Network Service Descriptors

Data Guard connect-time failover occurs when a connection request is forwarded to another listener if the connection fails. Connect-time failover is enabled by service registration, because the listener knows which available Oracle instance provides the requested service.

The following is an Oracle Net connection descriptor in the `tnsnames.ora` file:

```
sales.us.acme.com=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.acme.com)))
```

Note that the SALES net service name contains multiple address lists (two because it is a two-node cluster) for the production and standby clusters. The second address list enables connect-time failover if the first connection fails. This works for all protection modes.

To add a network protocol address to an existing net service name or database service, use either Oracle Enterprise Manager or Oracle Net Manager.

See Also: *Oracle Net Services Administrator's Guide*

Recommendations for Backup and Recovery

While it is prudent that every database have a good backup, recovery using a backup is not always the fastest solution. Other available Oracle technologies, such as RAC, Data Guard, and flashback technology often provide faster means of recovering from an outage than restoring from backups.

A good backup and recovery strategy is still vital to the overall high availability solution and ensures that specific outages are recovered from in an acceptable amount of time. The following topics are included in this section:

Use Recovery Manager to Back Up Database Files
Understand When to Use Backups
Use an RMAN Recovery Catalog

Use the Autobackup Feature for the Control File and SPFILE
Use Incrementally Updated Backups to Reduce Restoration Time
Enable Change Tracking to Reduce Backup Time
Create Database Backups on Disk in the Flash Recovery Area
Create Tape Backups from the Flash Recovery Area
Determine Retention Policy and Backup Frequency
Configure the Size of the Flash Recovery Area Properly
In a Data Guard Environment, Back Up to the Flash Recovery Area on All Sites
During Backups, Use the Target Database Control File as the RMAN Repository
Regularly Check Database Files for Corruption
Periodically Test Recovery Procedures
Back Up the OCR to Tape or Offsite

See Also:

- *Oracle Database Backup and Recovery Basics*
- *Oracle Database Backup and Recovery Advanced User's Guide* for more information about backup and recovery topics, including RMAN

Use Recovery Manager to Back Up Database Files

Recovery Manager (RMAN) uses server sessions to perform backup and recovery operations and stores metadata about backups in a repository. RMAN offers many advantages over typical user-managed backup methods, such as the ability to do online database backups without placing tablespaces in backup mode; support for incremental backups; data block integrity checks during backup and restore operations; and the ability to test backups and restores without actually performing the operation. RMAN automates backup and recovery, whereas the user-managed method requires you to keep track of all database files and backups. For example, instead of requiring you to locate backups for each datafile, copy them to the correct place using operating system commands, and choose which logs to apply; RMAN manages these tasks automatically. There are also capabilities of Oracle recovery that are only available when using RMAN, such as block media recovery.

Understand When to Use Backups

Most production database unscheduled outages are either handled automatically by various database components or are resolved by using another technology to restore a backup. For example, some outages are handled best by using Flashback

Database or the standby database. However, there are situations that require using database backups, including the following:

- Perform Regular Backups
- Initial Data Guard Environment Set-Up
- Recovering from Data Failures Using File or Block Media Recovery
- Double Failure Resolution
- Long-Term Backups

Perform Regular Backups

Perform regular backups.

> **See Also:** "Determine Retention Policy and Backup Frequency" on page 7-40

Initial Data Guard Environment Set-Up

During initial set-up of a standby database, a backup of the production database is required at the secondary site to create the initial standby database.

> **See Also:** *Oracle Data Guard Concepts and Administration*

Recovering from Data Failures Using File or Block Media Recovery

When a data failure, which includes block corruption and media failure, occurs in an environment that does not include Data Guard, the only method of recovery is using an existing backup. Even with Data Guard, the most efficient means of recovering from data failure may be restoring and recovering the affected object from an existing backup.

> **See Also:** "Recovery Methods for Data Failures" on page 10-32

Double Failure Resolution

A double failure scenario affects the availability of both the production and standby databases. The only resolution of this situation is to re-create the production database from an available backup, then re-create the standby database. An example of a double failure scenario is a site outage at the secondary site, which eliminates fault tolerance, followed by a media failure on the production database. Some multiple failures, or more appropriately disasters (such as a primary site outage followed by a secondary site outage) may require the use of backups that

exist in an offsite location, so developing and following a process to deliver and maintain backup tapes at an offsite location is necessary to restore service in the most dire of circumstances.

Long-Term Backups

Some businesses require the ability to maintain long-term backups that may be needed years into the future. By using RMAN with the KEEP option, it is possible to retain backups that are exempt from the retention policy and never expire, providing the capability to restore and recover the database to any desired point in time. It is important that a recovery catalog be used for the RMAN repository so that backup metadata is not lost due to lack of space, which may occur when using the target database control file for the RMAN repository.

Use an RMAN Recovery Catalog

RMAN automatically manages the backup metadata in the control file of the database that is being backed up. To protect and keep backup metadata for long periods of time, the RMAN repository, usually referred to as a recovery catalog, is created in a separate database. There are many advantages of using a recovery catalog, including the ability to store backup information long-term, the ability to store metadata for multiple databases, and the ability to restore an available backup on to another system. In addition, if you are using only the target database control file to house the repository, the control file, with its limited maximum size, may not be large enough to hold all desired backup metadata. If the control file is too small to hold additional backup metadata, then existing backup information is overwritten, making it difficult to restore and recover using those backups.

Use the Autobackup Feature for the Control File and SPFILE

RMAN can be configured to automatically back up the control file and server parameter file (SPFILE) whenever the database structure metadata in the control file changes and whenever a backup record is added. The autobackup enables RMAN to recover the database even if the current control file, catalog, and SPFILE are lost. The RMAN autobackup feature is enabled with the CONFIGURE CONTROLFILE AUTOBACKUP ON statement.

Use Incrementally Updated Backups to Reduce Restoration Time

Oracle's incrementally updated backups feature enables you to create an image copy of a datafile, then regularly create incremental backups of the database and apply them to that image copy. The image copy is updated with all changes up

through the SCN at which the incremental backup was taken. RMAN can use the resulting updated datafile in media recovery just as it would use a full image copy taken at that SCN, without the overhead of performing a full image copy of the database every day. A backup strategy based on incrementally updated backups can help minimize MTTR for media recovery.

See Also: *Oracle Database Backup and Recovery Basics*

Enable Change Tracking to Reduce Backup Time

Oracle's change tracking feature for incremental backups improves incremental backup performance by recording changed blocks in each datafile in a change tracking file. If change tracking is enabled, then RMAN uses the change tracking file to identify changed blocks for incremental backup, thus avoiding the need to scan every block in the datafile.

See Also: *Oracle Database Backup and Recovery Basics*

Create Database Backups on Disk in the Flash Recovery Area

Using automatic disk-based backup and recovery, you can create a flash recovery area, which automates management of backup-related files. Choose a location on disk and an upper bound for storage space and set a retention policy that governs how long backup files are needed for recovery. Oracle manages the storage used for backup, archived redo logs, and other recovery-related files for your database within that space. Files no longer needed are eligible for deletion when RMAN needs to reclaim space for new files.

See Also: *Oracle Database Backup and Recovery Basics*

Create Tape Backups from the Flash Recovery Area

Use the `BACKUP RECOVERY FILE DESTINATION` RMAN command to move disk backups created in the flash recovery area to tape. Tape backups are used for offsite and long-term storage and are used to handle certain outage scenarios.

See Also: *Oracle Database Backup and Recovery Basics*

Determine Retention Policy and Backup Frequency

The backup retention policy is the rule set regarding which backups must be retained (on disk or other backup media) to meet recovery and other requirements. It may be safe to delete a specific backup because it is old enough to be superseded

by more recent backups or because it has been stored on tape. You may also need to retain a specific backup on disk for other reasons such as archival requirements. A backup that is no longer needed to satisfy the backup retention policy is said to be **obsolete**.

Backup retention policy can be based on **redundancy** or a **recovery window**. In a redundancy-based retention policy, you specify a number *n* such that you always keep at least *n* distinct backups of each file in your database. In a recovery window-based retention policy, you specify a time interval in the past (for example, one week or one month) and keep all backups required to let you perform point-in-time recovery to any point during that window.

Frequent backups are essential for any recovery scheme. Base the frequency of backups on the rate or frequency of database changes such as:

- Addition and deletion of tables
- Insertions and deletions of rows in existing tables
- Updates to data within tables

The more frequently your database is updated, the more often you should perform database backups. If database updates are relatively infrequent, then you can make whole database backups infrequently and supplement them with incremental backups, which will be relatively small because few blocks have changed.

See Also: *Oracle Database Backup and Recovery Basics*

Configure the Size of the Flash Recovery Area Properly

Configuring the size of the flash recovery area properly enables fast recovery from user error with Flashback Database and fast recovery from data failure with file or block media recovery from disk. The appropriate size of the flash recovery area depends on the following: retention policy, backup frequency, size of the database, rate and number of changes to the database. Specific formulas for determining the proper size of the flash recovery area for different backup scenarios are provided in *Oracle Database Backup and Recovery Basics.*

In a Data Guard Environment, Back Up to the Flash Recovery Area on All Sites

Take backups at the primary and secondary sites. The advantages of this practice are as follows:

- It significantly reduces MTTR in certain double outage scenarios.

- It avoids introducing new backup procedures upon a switchover or failover.
- RMAN file and block media recovery is a recovery option for data failure outages at both primary and secondary sites.

Consider a scenario in which backups are done only at the secondary site. Suppose there is a site outage at the secondary site where the estimated time to recover is three days. The primary site is completely vulnerable to an outage that is typically resolved by a failover, also to any outage that could be resolved by having a local backup (such as a data failure outage resolved by block media recovery). In this scenario, a production database outage can be resolved only by physically shipping the off-site tape backups that were taken at the standby site. If primary site backups were available, then restoring locally would be an available option in place of the failover than cannot be done. Data may be lost, but having primary site backups significantly shortens the MTTR.

Another undesirable approach is to start taking primary site backups at the time that there is a secondary site outage. However, this approach should be avoided because it is introducing new processes and procedures at a time when the environment is already under duress and the impact of a mistake by staff will be magnified. Also, it is not a time to learn that backups cannot be taken at the primary site.

In addition, primary site disk backups are necessary to ensure a reasonable MTTR when using RMAN file or block media recovery. Without a local on-disk backup, a backup taken at the standby site must be restored to the primary site, significantly lengthening the MTTR for this type of outage.

During Backups, Use the Target Database Control File as the RMAN Repository

During backups, use the target database control file as the RMAN repository and resynchronize afterward with the RMAN `RESYNC CATALOG` command.

When creating backups to disk or tape, use the target database control file as the RMAN repository so that the ability to back up or the success of the backup does not depend on the availability of the RMAN catalog in the manageability database. This is accomplished by running RMAN with the `NOCATALOG` option. After the backup is complete, the new backup information stored in the target database control file can be resynchronized with the recovery catalog using the `RESYNC CATALOG` command.

Regularly Check Database Files for Corruption

Using the BACKUP VALIDATE RMAN command, database files should be checked regularly for block corruptions that have not yet been reported by a user session or by normal backup operations. RMAN scans the specified files and verifies content-checking for physical and logical errors but does not actually perform the backup or recovery operation. Oracle records the address of the corrupt block and the type of corruption in the control file. Access these records through the V$DATABASE_BLOCK_CORRUPTION view, which can be used by RMAN block media recovery.

If BLOCK CHANGE TRACKING is enabled, then do not use the INCREMENTAL LEVEL option with BACKUP VALIDATE to ensure that all data blocks are read and verified.

To detect all types of corruption that are possible to detect:

- Do not specify the MAXCORRUPT option
- Do not specify the NOCHECKSUM option
- Do specify the CHECK LOGICAL option

Periodically Test Recovery Procedures

Complete, successful, and tested backups are fundamental to the success of any recovery. Create test plans for the different outage types. Start with the most common outage types and progress to the least probable. Issuing backup procedures does not ensure that the backups are successful; they must be rehearsed. Monitor the backup procedure for errors, and validate backups by testing your recovery procedures periodically. Also, validate the ability to do backups and restores by using the RMAN commands BACKUP VALIDATE and RESTORE... VALIDATE commands.

See Also: *Oracle Database Backup and Recovery Basics*

Back Up the OCR to Tape or Offsite

The Oracle Cluster Registry (OCR) contains cluster and database configuration information for RAC and Cluster Ready Services (CRS), such as the cluster database node list, CRS application resource profiles, and Event Manager (EVM) authorizations. Using the ocrconfig tool, there are two methods of copying OCR content and using the content for recovery. The first method uses automatically generated physical OCR file copies. The second method uses manually created

logical OCR export files. The backup file created with `ocrconfig` should be backed as part of the operating system backup using standard operating system or third-party tools.

See Also: *Oracle Real Application Clusters Administrator's Guide*

Recommendations for Fast Application Failover

In any high availability architecture, client and mid-tier applications can be redirected to available services within a Real Application Cluster and with some customization to a Data Guard or replicated database. This redirection can usually be transparent and can be used to reduce or eliminate both planned and unplanned downtime to the client or mid-tier application.

Services are prerequisites for fast, transparent application failover. When you create services in RAC, you can assign the services to instances for preferred (normal) and available (recovery) processing. When an instance to which you have assigned a service becomes unavailable, RAC can reconnect users connected to that instance to an available instance without service interruptions. Clients and mid-tier applications make connection requests by specifying a service using a global name. The connection information must be aware of all potential production instances or databases that can publish that service. Services enable you to model and deploy both planned and unplanned operations for any type of high availability or disaster recovery scenario.

To respond to changes in the cluster database, RAC's Cluster Ready Services (CRS), event system, and service callouts can be used to notify clients and mid-tier applications automatically. Event notifications can be configured to initiate recovery processing after failures to eliminate network timeouts and to provide end-to-end control over essential resources. These rapid notifications are done automatically from RAC to JDBC clients through JDBC fast connection failover. However, RAC provides a robust callout and event system that enables the user to customize specialized callouts to respond to database `UP` and `DOWN` events. Use these callouts to notify middle-tier applications to interrupt existing problematic connections and redirect new connections to available resources.

For disaster recovery, the new production database can also be configured to publish the production service while stopping the services on the old production database. Again, callouts are required to notify the mid-tier applications.

To configure for fast application failover, follow these recommendations for middle-tier applications and clients:

- Configure Connection Descriptors for All Possible Production Instances

- Use RAC Availability Notifications and Events
- Use Transparent Application Failover If RAC Notification Is Not Feasible
  - Use proper retries and delays to connect to the disaster recovery site
  - Adjust TCP system parameters to reduce TCP/IP timeouts

Follow this recommendation for all databases:

- Configure Services

Follow these recommendation for RAC:

- Configure CRS for High Availability
- Configure Service Callouts to Notify Middle-Tier Applications and Clients

Follow these recommendations for Data Guard, replicated, or distributed environments:

- Publish Standby or Nonproduction Services
- Publish Production Services

Configure Connection Descriptors for All Possible Production Instances

Clients and mid-tier applications make connection requests by specifying a service using a global name. The connection information must be aware of all potential production instances or databases that are capable of publishing that service. Furthermore, these connection descriptors should ideally be stored in an LDAP or Oracle Name server for ease of maintenance and administration.

This section includes three sample Oracle Net connection descriptors. Use the PROD_RAC connection descriptor in Example 7–1when there is no standby database available or if DNS site failover is deployed.

The PROD_RAC_DG connection descriptor in Example 7–2 has an address list that contains all production RAC instances and the Data Guard standby instances. This example can be used in the case of a production database outage when the hardware cluster is still available. It helps you avoid TCP/IP timeouts.

When the entire hardware cluster fails, the connection needs to be manually adjusted to point to the standby database using the connection descriptor provided in Example 7–3.

For disaster recovery, client-side DNS failover or site failover is recommended over listing both production instances and standby database instances.

See Also: "Complete or Partial Site Failover" on page 10-3

Example 7–1 Connection Descriptor: No Standby Database Available or DNS Site Failover is Deployed

```
PROD_RAC=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (LOAD_BALANCE=on)
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE1)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE2)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE3)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE4)(PORT = 1520))
    (CONNECT_DATA =
      (SERVICE_NAME = MAA_PROD)))
```

Example 7–2 Connection Descriptor: Production Database Outage When Hardware Cluster is Available

```
PROD_RAC_DG=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (LOAD_BALANCE=on)
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE1)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE2)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE3)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE4)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE1)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE2)(PORT = 1520)))
    (CONNECT_DATA =
      (SERVICE_NAME = MAA_PROD)))
```

Example 7–3 Connection Descriptor: Hardware Cluster Fails

```
PROD_DR=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (LOAD_BALANCE=on)
      (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE1)(PORT = 1520))
           (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE2)(PORT = 1520)))
    (CONNECT_DATA =
      (SERVICE_NAME = MAA_PROD)))
```

Use RAC Availability Notifications and Events

Ideally, the middle-tier applications and clients should use the automatic RAC availability notifications and events. Applications that use Oracle Database 10*g* JDBC fast connection failover subscribe to these events automatically. Other applications may need to configure these service callouts and modify the applications to react to them.

See Also:

- "Configure Service Callouts to Notify Middle-Tier Applications and Clients" on page 7-50
- *Oracle Database JDBC Developer's Guide and Reference*

Use Transparent Application Failover If RAC Notification Is Not Feasible

If you cannot use RAC notification or if RAC is not deployed, then use Transparent Application Failover (TAF). When Oracle Call Interface (OCI) client applications are used, Transparent Application Failover (TAF) can be configured to transparently fail over connections between an application server and a database server.

OCI client applications can take advantage of automatic reconnection after failover and callback functions that help to automate state recovery. They can also replay interrupted SELECT statements and callback functions that help to automate state recovery. The Oracle JDBC and ODBC drivers also support automatic database reconnection and replay of interrupted SELECT statements without the need for any additional application coding.

The TAF configuration is specified in the connect string that clients use to connect to the database.

The following sample TAF connection descriptor is used to describe the impact of TAF and how to use each component.

```
PROD=
  (DESCRIPTION =
    (FAILOVER=on)
    (ADDRESS_LIST =
      (LOAD_BALANCE=on)
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE1)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE2)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE3)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE4)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE1)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE2)(PORT = 1520)))
```

```
    (CONNECT_DATA =
      (SERVICE_NAME = MAA_PROD)
 (FAILOVER_MODE =
(BACKUP=PROD_BACKUP)(TYPE=SESSION)(METHOD=BASIC)
(RETRIES=12)(DELAY=5))))

PROD_BACKUP=
  (DESCRIPTION =
    (FAILOVER=on)
    (ADDRESS_LIST =
      (LOAD_BALANCE=on)
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE1)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE2)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE3)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = RAC_INSTANCE4)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE1)(PORT = 1520))
      (ADDRESS = (PROTOCOL = TCP)(HOST = DG_INSTANCE2)(PORT = 1520)))
    (CONNECT_DATA =
      (SERVICE_NAME = MAA_PROD)
      (FAILOVER_MODE =
(BACKUP=PROD)(TYPE=SESSION)(METHOD=BASIC)
(RETRIES=12)(DELAY=5))))
```

New Connections

New connections use the address list and connect to the first available listener that has the service (MAA_PROD) registered. This is true for both instance and listener failure. If there is an attempt to connect to a failed node, then a TCP/IP timeout occurs. Retries and delay have no effect on new connections because new connections attempt the address list only once.

Existing Connections

Existing connections use the backup connection descriptor and wait the number of seconds specified by DELAY between each iteration. After attempting all addresses in the backup connection descriptor, the client waits the number of seconds specified by DELAY before attempting the address list again. The client retries the address list up to the number of times specified by RETRIES. If the service is not available anywhere after the number of seconds that is equal to RETRIES times DELAY, then the client receives an ORA-3113 error. The maximum switchover or failover times should be less than RETRIES*DELAY if you want automatic client failover to a disaster recovery site.

LOAD_BALANCE Parameter in the Connection Descriptor

LOAD_BALANCE sets the client-side load balancing. When it is set to ON, the client randomly selects an address from the address list. If a listener has multiple instances registered with it that offer the same service, then the listener can balance client requests between the instances based on the load on the instances at that time.

FAILOVER Parameter in the Connection Descriptor

Set FAILOVER to ON. The client fails through the address list if one or more of the services, instances, listeners, or nodes on the list is down or not available.

SERVICE_NAME Parameter in the Connection Descriptor

The service name is published by the database to the listener.

RETRIES Parameter in the Connection Descriptor

This parameter determines how many times an existing connection retries the addresses in the BACKUP list or after a failover. This parameter has no effect on new connections. New clients go through the address list only once.

DELAY Parameter in the Connection Descriptor

This parameter determines the number of seconds the client waits between each retry. After going through the address list, the client waits for the number of seconds specified by DELAY before retrying. There is no delay between individual addresses in the address list. The delay applies only after the whole list has been traversed.

With RAC, TCP/IP timeouts due to an unavailable node in a cluster are avoided because RAC manages a Virtual Internet Protocol (VIP) and cluster alias. However, TCP/IP timeouts cannot be avoided when the entire cluster or non-RAC host is not available. To avoid this TCP/IP timeout, the customer should do one of the following:

- Create a special event and callout to detect and react to such as event
- Adjust TCP/IP parameters to reduce overall timeout impact

The customized callout should interrupt existing connections and redirect new connections with a new connection descriptor that does not contain the unavailable nodes or clusters.

Adjusting TCP/IP parameters may have other application and system impact, so always use caution. However, the following TCP/IP parameters were modified on a Solaris platform to reduce overall TCP/IP timeout intervals in Oracle testing:

- `tcp_ip_abort_cinterval`
- `tcp_ip_abort_interval`
- `tcp_keepalive_interval`

Check your operating system platform documentation for similar parameters.

Configure Services

Within RAC, use the Database Configuration Assistant (DBCA), Server Control (`SRVCTL`), or the `DBMS_SERVICE` PL/SQL package to create services. Then use the DBCA or Enterprise Manager to administer them. If this is a non-RAC environment, then set the `SERVICE_NAME` database initialization parameter.

Configure CRS for High Availability

CRS supports services and the workload management framework that maintains continuous availability of the services. CRS also supports the other RAC resources such as the database, the database cluster aliases, and the resources that are local to every node that supports RAC.

Node resources include the virtual internet protocol (VIP) address for the node, the Global Services Daemon, the Enterprise Manager Agent, and the Oracle Net listeners. These resources are automatically started when CRS starts with the node and CRS automatically restarts them if they fail.

See Also:

- *Oracle Real Application Clusters Administrator's Guide*
- *Oracle Real Application Clusters Deployment and Performance Guide*

Configure Service Callouts to Notify Middle-Tier Applications and Clients

Configure service callouts to notify middle-tier applications and clients about `UP`, `DOWN`, and `NOT_RESTARTING` events. RAC automatically notifies JDBC clients through JDBC fast connection failover without any adjustments. In the rare case that the entire RAC cluster fails, a separate notification and callout is required to

notify the middle-tier applications to connect to a disaster recovery or secondary database.

If the middle-tier application or clients are not JDBC clients, then you must use RAC's event management and service callout facilities to configure a customized callout. The callout needs to notify the middle-tier application to do the following:

- Interrupt existing connections to problematic or unavailable nodes or instances
- Redirect new connections to available production instances or nodes

Interrupting existing connections helps avoid long TCP/IP timeout delays. Redirecting new connections to available production instances or nodes may require passing a new connection descriptor that does not include any inaccessible hosts so that TCP/IP timeouts can be avoided.

See Also: *Oracle Real Application Clusters Administrator's Guide*

Publish Standby or Nonproduction Services

When RAC and Enterprise Manager are integrated, standby or nonproduction services can be published automatically. If the standby database is not managed by Enterprise Manager or is not part of a RAC environment, then you can manually alter the SERVICE_NAME database initialization parameter to be a nonproduction service. For example:

```
SQL> ALTER SYSTEM SET SERVICE_NAME='STANDBY';
```

Publish Production Services

When RAC and Enterprise Manager are integrated, production services can be published automatically. If the new production database is not managed by Enterprise Manager or is not part of a RAC environment, then you can manually alter the SERVICE_NAME database initialization parameter to be set to different production services. For example:

```
SQL> ALTER SYSTEM SET SERVICE_NAME='PROD_SVC1, PROD_SVC2, PROD_SVC3';
```

PROD_SVC1 can be SALES or HR, for example.

Part IV

Managing a Highly Available Oracle Environment

This part describes how to manage a highly available Oracle environment.

This part contains the following chapters:

- Chapter 8, "Using Oracle Enterprise Manager for Monitoring and Detection"
- Chapter 9, "Recovering from Outages"
- Chapter 11, "Restoring Fault Tolerance"

8

Using Oracle Enterprise Manager for Monitoring and Detection

This chapter provides recommendations for using Oracle Enterprise Manager to monitor and maintain a highly available environment across all tiers of the application stack. In addition, it describes how to create an Enterprise Manager configuration that is highly available.

- Overview of Monitoring and Detection for High Availability
- Using Enterprise Manager for System Monitoring
- Managing the HA Environment with Enterprise Manager
- Highly Available Architectures for Enterprise Manager
- Additional Enterprise Manager Configuration

Overview of Monitoring and Detection for High Availability

Continuous monitoring of the system, network, database operations, application, and other system components ensures early detection of problems. Early detection improves the user's system experience because problems can be resolved faster. In addition, monitoring captures system metrics to indicate trends in system performance growth and recurring problems. This information can facilitate prevention, enforce security policies, and manage job processing. For the database server, a sound monitoring system needs to measure availability and detect events that can cause the database server to become unavailable and provide immediate notification to responsible parties for critical failures.

The monitoring system itself needs to be highly available and adhere to the same operational best practices and availability practices as the resources it monitors. Failure of the monitoring system leaves all systems that it monitors unable to capture diagnostic data or alert the administrator of problems.

Oracle Enterprise Manager provides the management and monitoring capabilities with many different notification options. This chapter provides recommendations for using Enterprise Manager to monitor and maintain a highly available environment across all tiers of the application stack. Recommendations are available for methods of monitoring the environment's availability and performance and for using the tools in response to changes in the environment. In addition, there is a description of how to create an Enterprise Manager configuration that is highly available, as well as additional configuration tips.

Using Enterprise Manager for System Monitoring

This section provides an overview of the concepts and facilities available in Enterprise Manager.

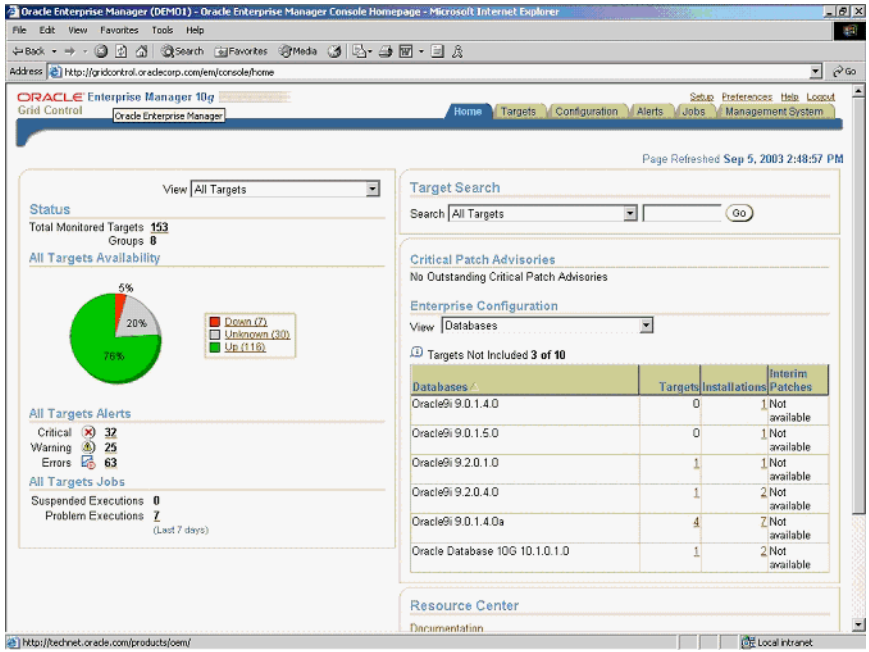
A major benefit of Enterprise Manager is its ability to manage components across the entire application stack from the host operating system to a user or packaged application. Enterprise Manager treats each of the layers in the application as a **target**. Targets, such as databases, application servers, and hardware, can then be viewed along with other targets of the same type or can be grouped together by application type. All targets can also be reviewed in a single view. Each target type has a default generated home page that displays a summary of relevant details for a specific target. Different types of targets can be grouped together by function, that is, as resources that support the same application.

Every target is monitored by an Oracle Management Agent. Every Management Agent runs on a machine and is responsible for a set of targets. The targets can be

on a machine that is different from the machine that the Management Agent is on. For example, a Management Agent can monitor a storage array that cannot host an agent natively. When a Management Agent is installed on a host, the host is automatically discovered along with other targets that are on the machine.

The Grid Control home page shown in Figure 8–1 provides a picture of the availability of all of the discovered targets.

Figure 8–1 Grid Control Home Page



The Grid Control home page shows the following major kinds of information:

- A snapshot of the current availability of all of the targets. The pie chart associated with availability gives the administrator an immediate indication of any target that is not available (Down) or has lost communication with the console (Unknown).
- An overview of how many alerts (for events) and problems (for jobs) are known in the entire monitored system. You can display detailed information by clicking the links or by navigating to **Alerts** from the upper right portion of any Enterprise Manager page.

- A target shortcut. This is intended for administrators who have to perform a task for a specific target.
- An overview of what is actually discovered in the system. This list can be shown at the hardware level and the Oracle level.
- A set of useful links to other Oracle online resources.

Alerts are generated by a combination of factors and are defined on specific **metrics**. A metric is a data point sampled by a Management Agent and sent to the Oracle Management Repository. It could be the availability of a component through a simple heartbeat test or an evaluation of a specific performance measurement such as "disk busy" or percentage of processes waiting for a specific wait event.

There are four states that can be checked for any metric: error, warning, critical, and clear. The administrator must make policy decisions such as:

- What objects should be monitored (databases, nodes, listeners, or other services)?
- What instrumentation should be sampled (such as availability, CPU percent busy)?
- How frequently should the event be sampled?
- What should be done when the metric exceeds a predefined threshold?

All of these decisions are predicated on the business needs of the system. For example, all components may be monitored for availability, but some systems may be monitored only during business hours. Systems with specific performance problems can have additional performance tracing enabled to debug a problem.

The rest of this section includes the following topics:

- Set Up Default Notification Rules for Each System
- Use Database Target Views to Monitor Health, Availability, and Performance
- Use Event Notifications to React to Metric Changes
- Use Events to Monitor Data Guard system Availability

See Also:

- *Oracle Enterprise Manager Concepts*
- *Oracle 2 Day DBA* for information about setting metrics with Oracle Enterprise Manager

Set Up Default Notification Rules for Each System

Notification Rules are defined sets of alerts on metrics that are automatically applied to a target when it is discovered by Enterprise Manager. For example, an administrator can create a rule that monitors the availability of database targets and generates an e-mail message if a database fails. After that rule is generated, it is applied to all existing databases and any database created in the future. Access these rules by navigating to **Preferences** and then choosing **Rules**.

The rules monitor problems that require immediate attention, such as those that can affect service availability and Oracle or application errors. Service availability can be affected by an outage in any layer of the application stack: node, database, listener, and critical application data. A service availability failure, such as the inability to connect to the database, or the inability to access data critical to the functionality of the application, must be identified, reported, and reacted to quickly. Potential service outages such as a full archive log directory also need to be addressed correctly to avoid a system outage.

Enterprise Manager provides a series of default rules that provide a strong framework for monitoring availability. A default rule is provided for each of the preinstalled target types that come with Enterprise Manager. These rules can be modified to conform to the policies of each individual site, and new rules can be created for site-specific targets or applications. The rules can also be set to notify users during specific time periods to create an automated coverage policy.

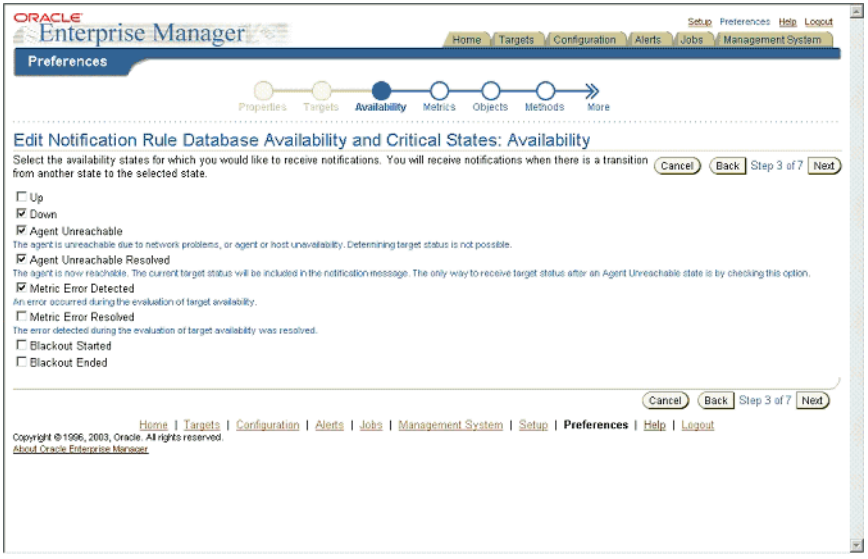
Consider the following recommendations:

- Modify each rule for high-value components in the target architecture to suit the required availability requirements by using the rules modification wizard. For the database rule, set the events in Table 8–1, Table 8–2, and Table 8–3 for each target. The frequency of the monitoring is determined by the service level agreement (SLA) for each component.
- Add Notification Methods and use them in each Notification Rule. By default, the easiest method for alerting an administrator to a potential problem is to send e-mail. Supplement this notification method by adding a callout to an SNMP trap or operating system script that sends an alert by some method other than e-mail. This avoids the problem that might occur if a component of the e-mail system has failed. Set additional Notification Methods by using the **Set-up** link at the top of any Enterprise Manager page.
- Modify Notification Rules to notify the administrator when there are errors in computing target availability. This may generate a false positive reading on the

availability of the component, but it ensures the highest level of notification to system administrators.

Figure 8–2 shows the Notification Rule property page for choosing availability states. Down, Agent Unreachable, Agent Unreachable Resolved, and Metric Error Detected are chosen.

Figure 8–2 Setting Notification Rules for Availability



In addition, modify the metrics monitored by the database rule to report the metrics shown in Table 8–1, Table 8–2, and Table 8–3. This ensures that these metrics are captured for all database targets and that trend data will be available for future analysis. All of the events described in Table 8–1, Table 8–2, and Table 8–3 can be accessed from the **Database Homepage**. Choose **All Metrics** > **Expand All**.

Space management conditions that have the potential to cause a service outage should be monitored using the events shown in Table 8–1.

Table 8–1 Recommendations for Monitoring Space

Metric	Recommendation
Tablespace Space Used (%)	Set this metric to monitor root file systems for any critical hardware server. This metric enables the administrator to choose the threshold percentages that Enterprise Manager tests against, as well as the number of samples that must occur in error before a message is generated to the administrator. The recommended default settings are 70 percent for a warning and 90 percent for an error, but these should be adjusted depending on system usage. This metric can be customized to monitor only specific tablespaces. This metric and similar events can be set in the Tablespace Full metric group.
Archiver Hung Alert Log Error	Set this metric to monitor the alert log for ORA-00257 errors, which indicate a full archive log directory. This metric can be set in the Alert Log Error Status metric group.
Archive Area Used(%)	Set this metric with thresholds and an appropriate sampling time. This metric can alert the administrator about a full archive directory, which can stop the system. The recommended default settings are 70 percent for a warning and 90 percent for an error, but these should be adjusted depending on system usage. This metric can be set in the Archive Area metric group.
Dump Area Used (%)	Set this metric to monitor the dump directory destinations. Dump space must be available so that the maximum amount of diagnostic information is saved the first time an error occurs. The recommended default settings are 70 percent for a warning and 90 percent for an error, but these should be adjusted depending on system usage. This metric can be set in the Dump Area metric group.

From the Alert Log Metric group, set Enterprise Manager to monitor the alert log for errors as shown in Table 8–2.

Table 8–2 Recommendation for Monitoring the Alert Log

Metric	Recommendation
Alert	Set this metric to send an alert when an ORA-6XX, ORA-1578 (database corruption), or ORA-0060 (deadlock detected) error occurs. If any other error is recorded, then a warning message is generated.
Data Block Corruption	Set this metric to monitor the alert log for ORA-01157 and ORA-27048 errors. They signal a corruption in an Oracle Database datafile.
Data Guard Log Transport	Set this metric.

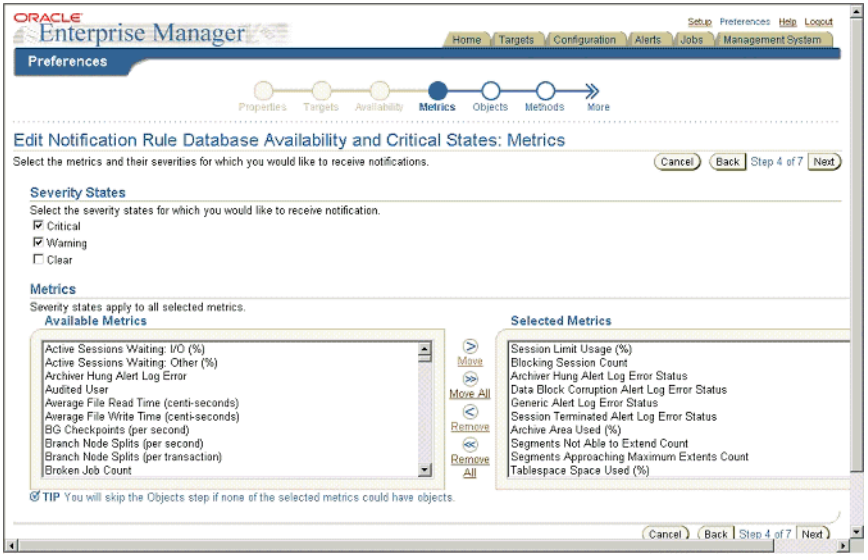
Monitor the system to ensure that the processing capacity is not exceeded. The warning and critical levels for these events should be modified based on the usage pattern of the system. Set the events from the Database Limits metric group. Table 8–3 contains the recommendations.

Table 8–3 Recommendations for Monitoring Processing Capacity

Metric	Recommendation
Process limit	Set thresholds for this metric to warn if the number of current processes approaches the value of the PROCESSES initialization parameter.
Session limit	Set thresholds for this metric to warn if the instance is approaching the maximum number of concurrent connections allowed by the database.

Figure 8–3 shows the Notification Rule property page for setting choosing metrics. The user has chosen Critical and Warning as the severity states for notification. The list of Available Metrics is shown in the left list box. The metrics that have been selected for notification are shown in the right list box.

Figure 8–3 Setting Notification Rules for Metrics

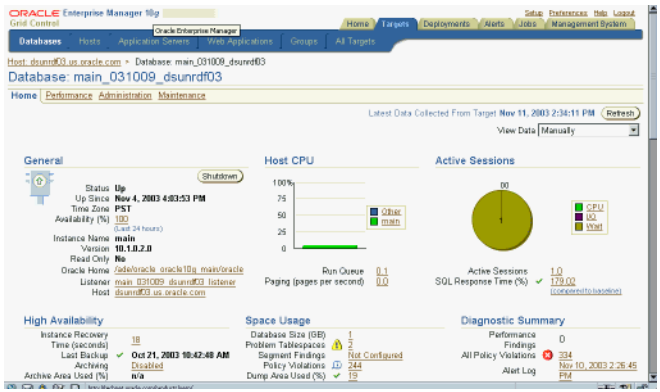


See Also: *Oracle 2 Day DBA* for information about setting up notification rules and metric thresholds

Use Database Target Views to Monitor Health, Availability, and Performance

The Database Targets page in Figure 8–4 shows an overview of system performance, space utilization, and the configuration of important availability components like archived redo log status, flashback log status, and estimated instance recovery time. Alerts are displayed immediately. Each of the alert values can be configured from links on this page

Figure 8–4 Overview of System Performance



Many of the metrics from the Enterprise Manager pertain to performance. A system without adequate performance is not an HA system, regardless of the status of any of the individual components. While performance problems seldom cause a major system outage, they can still cause an outage to a subset of customers. Outages of this type are commonly referred to as **application service brownouts**. The primary cause of brownouts is the intermittent or partial failure of one or more infrastructure components. IT managers must be aware of how the infrastructure components are performing (their response time, latency, and availability) and how they are affecting the quality of application service delivered to the end user.

A performance baseline, derived from normal operations that meet the SLA, should determine what constitutes a performance metric alert. Baseline data should be collected from the first day that an application is in production and should include the following:

- Application statistics (transaction volumes, response time, web service times)
- Database statistics (transaction rate, redo rate, hit ratios, top 5 wait events, top 5 SQL transactions)

- Operating system statistics (CPU, memory, I/O, network)

You can use Enterprise Manager to capture a snapshot of database performance as a baseline. Enterprise Manager compares these values against system performance and displays the result on the database Target page. It can also send alerts if the values deviate too far from the established baseline.

Set the database notification rule to capture the metrics listed in Table 8–4 for all database targets. Analysis of these parameters can then be done using one tool and historical data will be available.

Table 8–4 Recommended Notification Rules for Metrics

Metric	Recommendation
Disk I/O per Second	This is a database-level metric that monitors I/O operations done by the database. It sends an alert when the number of operations exceeds a user-defined threshold. Use this metric with operating system-level events that are also available with Enterprise Manager. Set this metric based on the total I/O throughput available to the system, the number of I/O channels available, network bandwidth (in a SAN environment), the effects of the disk cache if you are using a storage array device, and the maximum I/O rate and number of spindles available to the database.
% CPU Busy	Set this metric to warn at 75 percent and to show a critical alert between 85 percent and 90 percent. This usage may be normal at peak periods, but it may also be an indication of a runaway process or of a potential resource shortage.
% Wait Time	Excessive idle time indicates that a bottleneck for one or more resources is occurring. Set this metric based on the system wait time when the application is performing as expected.
Network Bytes per Second	This metric reports network traffic that Oracle generates. It can indicate a potential network bottleneck. Set this metric based in actual usage during peak periods.
Total Parses per Second	This metric measures SQL performance. It can indicate an application change or change in usage that has created a shortage of resources. Set it based on peak periods.

See Also: *Oracle Database Performance Tuning Guide* for more information about performance monitoring

Use Event Notifications to React to Metric Changes

There are many operating system events that can be used to supplement a suggested metric. Such operating system events are not required for each host and instance. All metrics defined here can be set individually by instance or database

using the **Manage Metrics** link at the bottom of the navigation bar of the object target page. The values that trigger a warning or critical alert can be changed here, and an operating system script can be activated to respond to an metric threshold, in addition to the standard alert being generated to the Oracle Enterprise Manager 10*g* Grid Control.

Use Events to Monitor Data Guard system Availability

Set Enterprise Manager metrics to monitor the availability of logical and physical Data Guard configurations. If a Data Guard environment is registered with the Data Guard Manager extension of Enterprise Manager, then set the events shown in

Table 8–5 Recommendations for Setting Data Guard Events

Metric	Recommendation
Data Guard Status	Set this metric to be notified of system problems in a Data Guard configuration.
Data Not Applied	Set this metric to be notified when the gap (measured in minutes) between the last archived redo log received and the last log applied on the standby database exceeds a user-defined threshold. This information can be used to warn the administrator if the recovery time for a standby instance will exceed the defined outage recovery service level. Set this metric based on the specifications for log application for the standby database.
Data Not Received	Set this metric to be notified if there is an extended delay in moving archived redo logs from the production database to the standby database. This metric occurs when the difference between the number of archived redo logs on the production database and the number of archived redo logs shipped to the standby site exceeds a user-defined threshold. The threshold should be based on the amount of time it takes to transport an archived redo log across the network. Set the sample time for the metric to be approximately the log transport time, and set the number of occurrences to be 2 or greater to avoid false positives. Recommended starting values for the warning and critical thresholds are 1 and 2.

Managing the HA Environment with Enterprise Manager

Use Enterprise Manager as a proactive part of administering any system as well as for problem notification and analysis. This section includes the following recommendations:

- Check Enterprise Manager Policy Violations
- Use Enterprise Manager to Manage Oracle Patches and Maintain System Baselines

- Use Enterprise Manager to Manage Data Guard Targets

See Also: *Oracle Enterprise Manager Administrator's Guide*

Check Enterprise Manager Policy Violations

Enterprise Manager comes with a pre-installed set of policies and recommendations of best practices for all databases. These policies are checked by default, and the number of violations is displayed on the Targets page in Figure 8–4. Select **Policy Violations** from the Targets page to see a list of all violations.

Use Enterprise Manager to Manage Oracle Patches and Maintain System Baselines

You can use Enterprise Manager to download and manage patches from `http://metalink.oracle.com` for any monitored system in the application environment. A job can be set up to routinely check for patches that are relevant to the user environment. Those patches can be downloaded and stored directly in the Management Repository. Patches can be staged from the Management Repository to multiple systems and applied during maintenance windows.

You can examine patch levels for one machine and compare them between machines in either a one-to-one or one-to-many relationship. In this case, a machine can be identified as a baseline and used to demonstrate maintenance requirements in other machines. This can be done for operating system patches as well as database patches.

Use Enterprise Manager to Manage Data Guard Targets

Enterprise Manager can be used to set up logical and physical standby databases for any database target. It also provides the ability to manage switchover and failover of database targets other than the database that contains the Management Repository.

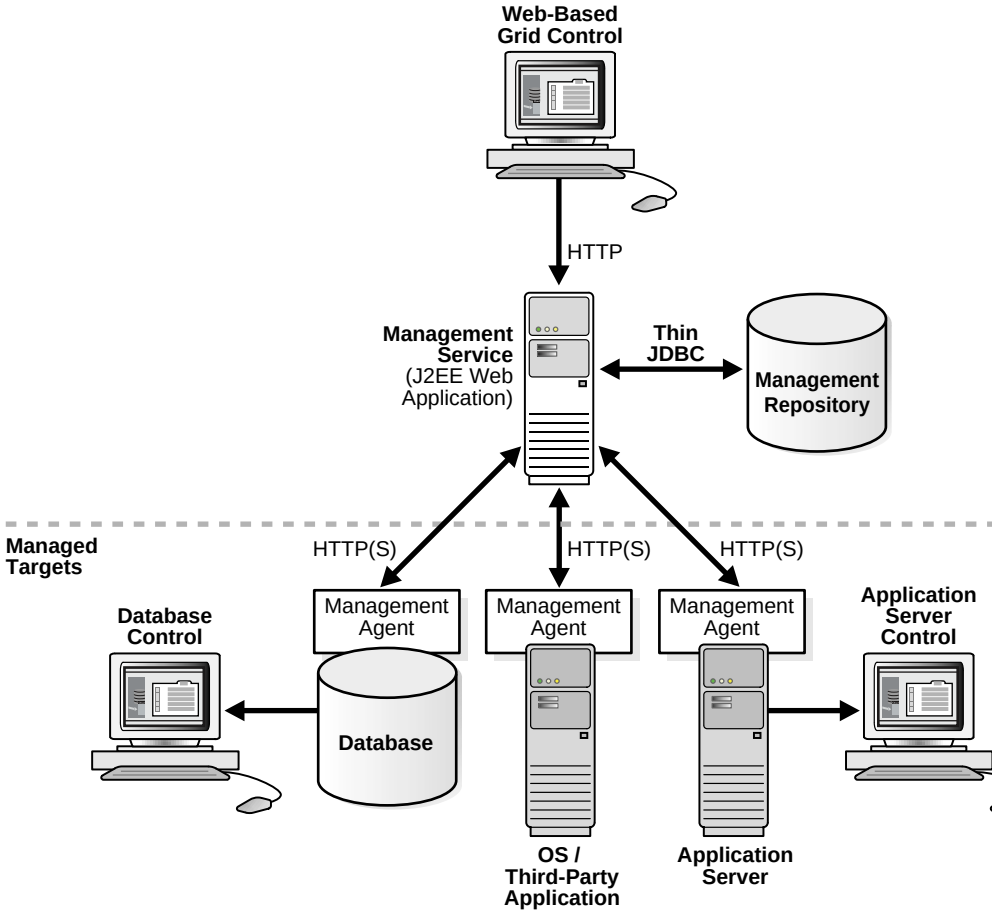
Enterprise Manager can also be used to monitor the health of a Data Guard configuration at a glance. From any database target page, navigate to the Data Guard status section by using the link in the High Availability section. The page shows the active standby databases for the primary target, the amount of log data waiting for shipment and receipt by the standby database and the data protection mode. You can also modify the data protection mode from this page.

This page contains a link to the **Verify** function, which checks the Data Guard environment and log transport services and displays warnings and errors. The Verify function must be run manually; it is not automatic.

Highly Available Architectures for Enterprise Manager

The Enterprise Manager architecture consists of a three-tier framework as shown in Figure 8–5.

Figure 8–5 Enterprise Manager Architecture



The components of the architecture are as follows:

- Web-based Grid Control: The Enterprise Manager user interface for centrally managing the entire computing environment from one location. All of the services within the enterprise, including hosts, databases, listeners, application

servers, HTTP Servers and Web applications are easily managed as one cohesive unit.

- Oracle Management Service and Oracle Management Repository: The Management Service is a J2EE Web application that renders the user interface for the Grid Control, works with all Management Agents in processing monitoring and job information, and uses the Management Repository as its data store. The Management Repository consists of tablespaces in an Oracle database that contain information about administrators, targets, and applications that are managed within Enterprise Manager.
- Oracle Management Agents: Management Agents are processes that are deployed on each monitored host. The Management Agent is responsible for monitoring all targets on the host, for communicating that information to the Management Service, and for managing and maintaining the host and the products installed on the host. The managed targets in the figure include the database, the third-party application, and the application server.
- The Database Control enables you to monitor and administer a single Oracle Database instance or a clustered database.
- The Application Server Control enables you to monitor and administer a single Oracle Application Server instance, a farm of Oracle Application Server instances, or Oracle Application Server Clusters.

Enterprise Manager provides a detailed set of tools to monitor itself.

The Management System page is a predefined component of Enterprise Manager that shows the administrator an overview of the Enterprise Manager components, backlogs in processing agent data, and component availability.

The Management System page shows essential metrics, including the amount of space left in the repository and the amount of data waiting to be loaded to the repository. This page also provides a view of alerts or warnings against the management system. The Repository Operations page provides an overview of the individual component tasks that make up the management system. The Repository Operations page shows the individual components at a glance, including the amount of CPU resource consumed and processing errors. A default notification rule is created when the product is installed and should be configured to notify the system administrator of a problem with any Enterprise Manager component.

Set the following options to monitor an Enterprise Manager environment:

- Modify the Repository Operations Notification rule to provide updates on Management Service Status, Targets Not Providing Data, and Total Loader Run

Time. Access this rule from the Notifications Rules page. See "Set Up Default Notification Rules for Each System" on page 8-5.

- Update the `emd.properties` with a valid e-mail address and mail server for any agent that monitors an Management Service or Management Repository node. This provides Enterprise Manager an additional method of notification if the repository fails. Instructions for setting `emd.properties` are in the Tip section of the Grid Control home page.

The rest of this section includes the following topics:

- Recommendations for an HA Architecture for Enterprise Manager
- Unscheduled Outages for Enterprise Manager

Recommendations for an HA Architecture for Enterprise Manager

The following recommendations are described in this section:

- Protect the Repository and Processes As Well as the Configuration They Monitor
- Place the Management Repository in a RAC Instance and Use Data Guard
- Configure At Least Two Management Service Processes and Load Balance Them
- Consider Hosting Enterprise Manager on the Same Hardware as an HA System
- Monitor the Network Bandwidth Between Processes and Agents

Protect the Repository and Processes As Well as the Configuration They Monitor

Availability requirements need to be addressed for each layer of the Enterprise Manager stack. The minimum recommendation for the Enterprise Manager repository and processes is to host them in a configuration that has the same protections as the system with the highest level of availability monitored by Enterprise Manager. The Enterprise Manager architecture must be as reliable as the application architecture. It is crucial for the monitoring framework to detect problems and manage repair as efficiently as possible. The Enterprise Manager implementation should be designed to be as available as the most available application it monitors because the Enterprise Manager framework is used to generate alerts if any monitored application fails.

Place the Management Repository in a RAC Instance and Use Data Guard

The Management Repository is the foundation of all Enterprise Manager operations. If the Enterprise Manager system is being used to monitor and alert on a system using a RAC and Data Guard configuration, but the Management Repository is hosted only on a single instance, then an outage of the Enterprise Manager system puts the administrator at risk of not being notified in a timely fashion of problems in production systems. Consider placing the Management Repository in a RAC instance to protect from individual instance failure and using Data Guard to protect from site failure.

Configure At Least Two Management Service Processes and Load Balance Them

For the middle tier, the baseline recommendation is to have a minimum of two Management Service processes, using a hardware server load balancer to mask the location of an individual Management Service process and a failure of any individual component. This provides immediate coverage for a single failure in the most critical components in the Enterprise Manager architecture with little interruption of service for all systems monitored using Enterprise Manager. Hardware server load balancers can also be monitored and configured using Enterprise Manager, providing coverage across the operating system stack. Management Service processes connect to the repository instances using Oracle Net.

Consider Hosting Enterprise Manager on the Same Hardware as an HA System

To reduce hardware overload and use current resources, the repository and Management Service processes can be hosted on the same hardware as another highly available production system. This assumes that the secondary site has the capacity and bandwidth to handle the production load plus an active Enterprise Manager repository and Management Service process. A hardware service load balancer should be used as a front end for multiple Management Service processes to manage failure of an individual Management Service and to balance the workload across the middle tier.

Agents from any monitored node in the environment can connect to any active Management Service processes. Load balancing of the agent processes connecting to the Management Service processes is handled internally by Enterprise Manager.

Monitor the Network Bandwidth Between Processes and Agents

Sufficient network bandwidth must be available to support the communication between the Management Service processes and the Management Agents. If the repository is used to manage a larger enterprise, then communication between

agents and Management Service processes can be significant, depending on the number of scheduled events and jobs. If the Enterprise Manager framework is used to monitor multiple applications and more dedicated system resources are required, then consider scaling the Management Repository and Management Service processes with additional nodes. The Management Repository and Management Service processes can be scaled independently. If required, additional hardware outside of the cluster can be added to scale the number of Management Service processes.

Unscheduled Outages for Enterprise Manager

Enterprise Manager is the primary control interface for managing your data center. An outage of Enterprise Manager causes a critical lack of visibility into the performance and availability metrics that allow the DBA to manage overall system performance. Table 8–6 describes the outages that can occur to any of the tiers involved in Enterprise Manager and how to recover from each outage.

Table 8–6 Unscheduled Outages for Enterprise Manager

Type of Outage	Possible Reasons for Outage	Solutions or Alternatives
Management Repository instance failure	Hardware failure, Oracle database failure, network failure to a single node of a RAC instance on the primary site, listener failure	This is best managed by using a RAC environment for the Management Repository. In a RAC environment, connections reconnect to the second node using Oracle Net failover. When the failed node is restored, the load is rebalanced automatically.
Primary site failure	Network outage to both nodes, cluster failure, interconnect failure, hardware failure	Requires Data Guard failover to secondary site: ■ Stop the listeners configured for Enterprise Manager traffic on the primary site nodes ■ Perform a Data Guard failover for the Management Repository ■ Start the Enterprise Manager listeners and Management Service processes on the new production site **Note:** This cannot be managed by Enterprise Manager directly. See "Data Guard Failover Using SQL*Plus" on page 10-11.
Management Agent failure	Process failure, accidental user termination	The Management Agent watchdog process restarts the Management Agent. The number of restarts is bounded by user-configurable parameters to avoid unnecessary processing on the monitored node.

Table 8–6 Unscheduled Outages for Enterprise Manager (Cont.)

Type of Outage	Possible Reasons for Outage	Solutions or Alternatives
Watchdog failure	Process failure, accidental user termination	No data is reported. Logging stops for the Management Agent. Any hanging processes must be manually stopped, and the Management Agent must be restarted. **Note:** The watchdog failure is not reported back to the Enterprise Manager GUI.
System state data is deleted or corrupted	Agent failure, user deletion of state files	Stop Management Agent processes (`EMAGENT` and `EMWD`) that are still running. Restart the agent (`emctl start agent`).
Management Service process failure	Process failure, accidental user termination	The Oracle Process Manager and Notification Server (OPMN) restarts the Management Service. A server load balancer (SLB) can be used for multiple Management Service processes. This masks process failures and distributes the workload across the middle tier. Failure of a Management Service causes the GUI session connected to it to fail. The GUI session must be restarted on a surviving Management Service.
Oracle Process Manager and Notification Server (OPMN) failure	Process failure, accidental user termination	No data is reported; logging stops for the Management Service process. Hanging processes need to be manually killed (on UNIX platforms) and the agent needs to be restarted. **Note:** Death of the watchdog is not reported back to the Enterprise Manager GUI.
Grid Control disconnect	Grid Control loses connection to Management Service because of a network problem, Management Service failure, or node failure	Because the Grid Control is stateless, it receives data from Management Service processes. The failure is resolved by connecting to a surviving Management Service or by starting the Grid Control itself.

Additional Enterprise Manager Configuration

This section contains additional configuration information that will be helpful in building Enterprise Manager in an MAA environment. It includes the following topics:

- Configure a Separate Listener for Enterprise Manager

- Install the Management Repository Into an Existing Database

Configure a Separate Listener for Enterprise Manager

Traffic from the Management Agents is routed to the Management Service processes and then to the Management Repository by Oracle Net. To isolate this traffic from other application traffic and to support Data Guard if required for site failover, configure the Enterprise Manager traffic through a separate listener. The listener is active only on the node where the active Enterprise Manager instance is running. Do not set the GLOBAL_DBNAME parameter in the listener.ora file because setting it disables Transparent Application Failover (TAF) and connect-time failover. Configure the LOCAL_LISTENER and REMOTE_LISTENER initialization parameters to enable dynamic service registration and cross-registration. The following is an example of a listener configuration:

```
LISTENER_N1=
        (description_list=
             (description=
               (address_list=
                (address=(protocol=tcp)(port=1521)(host=EMPRIM1.us.oracle.com))
               )
             )
        )
SID_LIST_LISTENER_N1 =
  (SID_LIST =
    (SID_DESC =
      (ORACLE_HOME = /mnt/app/oracle/product/10g)
      (SID_NAME = EM1)
    )
  )

LISTENER_DG=
        (description_list=
             (description=
               (address_list=
                (address=(protocol=tcp)(port=1529)(host=EMPRIM1.us.oracle.com))
               )
             )
        )
```

Install the Management Repository Into an Existing Database

To avoid installation problems when building a RAC-based repository, it is easier to install the Enterprise Manager into an existing database and build any tablespaces

in advance. Certain versions of the Oracle Universal Installer do not handle installing the repository into a RAC database. Build the database first; then use the Install option to install into an existing database.

9

Recovering from Outages

This chapter describes scheduled and unscheduled outages and the Oracle recovery process and architectural framework that can manage each outage and minimize downtime. This chapter contains the following sections:

- Recovery Steps for Unscheduled Outages
- Recovery Steps for Scheduled Outages

Recovery Steps for Unscheduled Outages

Unscheduled outages are unanticipated failures in any part of the technology infrastructure that supports the application, including the following components:

- Hardware such as host machines, storage, switches, cables, and cards
- Software, including the operating system, the Oracle database and application server, and application code
- Network infrastructure
- Naming services infrastructure
- Front-end load balancers
- The current production site

The monitoring and HA infrastructure should provide rapid detection and recovery from failures. Detection is described in Chapter 8, "Using Oracle Enterprise Manager for Monitoring and Detection", while this chapter focuses on the recovery operations for each outage.

Table 9–1 describes the unscheduled outages that impact the primary or secondary site components.

Table 9–1 Unscheduled Outages

Outage	Description	Examples
Site failure	The entire site where the current production database resides is unavailable. This includes all tiers of the application.	■ Disaster at the production site such as a fire, flood, or earthquake ■ Power outages. (If there are multiple power grids and backup generators for critical systems, then this should affect only part of the data center.)
Node failure	A node of the RAC cluster is unavailable or fails	■ A database tier node fails or has to be shut down because of bad memory or bad CPU ■ The database tier node is unreachable ■ Both of the redundant cluster interconnects fail, resulting in another node taking ownership

Table 9–1 Unscheduled Outages (Cont.)

Outage	Description	Examples
Instance failure	A database instance is unavailable or fails	An instance of the RAC database on the data server fails because of a software bug or an operating system or hardware problem
Clusterwide failure	The whole cluster hosting the RAC database is unavailable or fails. This includes failures of nodes in the cluster as well as any other components that result in the cluster being unavailable and the Oracle database and instances on the site being unavailable.	■ The last surviving node on the RAC cluster fails and cannot be restarted ■ Both of the redundant cluster interconnects fail ■ Database corruption is severe enough to disallow continuity on the current data server ■ Disk storage fails
Data failure	This failure results in unavailability of parts of the database because of media corruptions, inaccessibility, or inconsistencies.	■ A datafile is accidentally removed or is unavailable ■ Media corruption affects blocks of the database ■ Oracle block corruption is caused by operating system or other node-related problems
User error	This failure results in unavailability of parts of the database and causes transactional or logical data inconsistencies. It is usually caused by the operator or by bugs in the application code. This is estimated to be the greatest single cause of downtime.	Localized damage (needs surgical repair) ■ User error results in a table being dropped or in rows being deleted from a table Widespread damage (needs drastic action to avoid downtime) ■ Application errors result in logical corruptions in the database ■ Operator error results in a batch job being run more times than specified. **Note:** This category focuses on user errors that affect database availability and, in particular, cause transactional or logical data inconsistencies.

The rest of this section provides outage decision trees for unscheduled outages on the primary site and the secondary site. The decision trees appear in the following sections:

- Recovery Steps for Unscheduled Outages on the Primary Site
- Recovery Steps for Unscheduled Outages on the Secondary Site

The high-level recovery steps for each outage are listed with links to the detailed descriptions for each recovery step. These descriptions are found in Chapter 10, "Detailed Recovery Steps".

Some outages require multiple recovery steps. For example, when a site failure occurs, the outage decision matrix states that Data Guard failover must occur before site failover. Some outages are handled automatically without any loss of availability. For example, instance failure is managed automatically by RAC. Multiple recovery options for each outage are listed wherever relevant.

Recovery Steps for Unscheduled Outages on the Primary Site

If the primary site contains the production database and the secondary site contains the standby database, then the outages on the primary site are the ones of most interest. Solutions for these outages are critical for maximum availability of the system. Only the "Data Guard only" and MAA architectures have a secondary site to protect from site disasters. The estimated recovery times (ERT) are strictly examples derived from customer and actual testing experiences and do not reflect a guaranteed recovery time.

Table 9–2 summarizes the recovery steps for unscheduled outages on the primary site.

Table 9–2 Recovery Steps for Unscheduled Outages on the Primary Site

Reason for Outage	Recovery Steps for "Database Only" Architecture	Recovery Steps for "RAC Only" Architecture	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Site failure	ERT: hours to days 1. Restore site. 2. Restore from tape backups. 3. Recover database.	ERT: hours to days 1. Restore site. 2. Restore from tape backups. 3. Recover database.	ERT: minutes to an hour 1. Database Failover 2. Complete or Partial Site Failover	ERT: minutes to an hour 1. Database Failover 2. Complete or Partial Site Failover

Table 9–2 Recovery Steps for Unscheduled Outages on the Primary Site (Cont.)

Reason for Outage	Recovery Steps for "Database Only" Architecture	Recovery Steps for "RAC Only" Architecture	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Node failure	ERT: minutes to an hour 1. Restart node and restart database. 2. Reconnect users.	ERT: seconds to minutes Managed automatically by RAC Recovery	ERT: minutes to an hour 1. Restart node and restart database. 2. Reconnect users. *or* ERT: minutes to an hour 1. Database Failover 2. Complete or Partial Site Failover	ERT: seconds to minutes Managed automatically by RAC Recovery
Instance failure	ERT: minutes 1. Restart instance. 2. Reconnect users.	ERT: seconds to minutes Managed automatically by RAC Recovery	ERT: minutes 1. Restart instance. 2. Reconnect users.	ERT: seconds to minutes Managed automatically by RAC Recovery
Clusterwide failure	N / A	ERT: hours to days 1. Restore cluster or restore at least one node. 2. Restore from tape backups. 3. Recover database.	N / A	ERT: minutes to an hour 1. Database Failover 2. Complete or Partial Site Failover

Table 9–2 Recovery Steps for Unscheduled Outages on the Primary Site (Cont.)

Reason for Outage	Recovery Steps for "Database Only" Architecture	Recovery Steps for "RAC Only" Architecture	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Data failure	ERT: minutes to an hour Recovery Solutions for Data Failures	ERT: minutes to an hour Recovery Solutions for Data Failures	ERT: minutes to an hour Recovery Solutions for Data Failures *or* ERT: minutes to an hour 1. Database Failover 2. Complete or Partial Site Failover **Note:** For primary database media failures or media corruptions, database failover may minimize data loss.	ERT: minutes to an hour Recovery Solutions for Data Failures *or* ERT: minutes to an hour 1. Database Failover 2. Complete or Partial Site Failover **Note:** For primary database media failures or media corruptions, database failover may minimize data loss.
User error	ERT: minutes Recovering from User Error with Flashback Technology	ERT: minutes Recovering from User Error with Flashback Technology	ERT: minutes Recovering from User Error with Flashback Technology	ERT: minutes Recovering from User Error with Flashback Technology

Recovery Steps for Unscheduled Outages on the Secondary Site

Outages on the secondary site do not directly affect availability because the clients always access the primary site unless there is a switchover or failover. Outages on the secondary site may impact the MTTR if there are concurrent failures on the primary site. For most cases, outages on the secondary site can be managed with no impact on availability. However, if maximum protection mode is part of the configuration, then an unscheduled outage on the last surviving standby database causes downtime on the production database. After downgrading the data protection mode, you can restart the production database.

Table 9–3 summarizes the recovery steps for unscheduled outages of the standby database on the secondary site.

Table 9–3 Recovery Steps for Unscheduled Outages of the Standby Database on the Secondary Site

Reason for Outage	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Standby apply instance failure	1. Restart node and standby instance. 2. Restart recovery. If there is only one standby database and if maximum database protection is configured, then the production database will shut down to ensure that there is no data divergence with the standby database.	ERT: seconds Apply Instance Failover There is no effect on production availability if the production database Oracle Net descriptor is configured to use connect-time failover to an available standby instance. Restart node and instance when they are available.
Standby non-apply instance failure	N / A	There is no effect on availability because the primary node or instance receives redo logs and applies them with the recovery process. The production database continues to communicate with this standby instance. Restart node and instance when they are available.
Data failure such as media failure or disk corruption	Restoring Fault Tolerance after a Standby Database Data Failure	Restoring Fault Tolerance after a Standby Database Data Failure
Primary database resets logs because of flashback operations or media recovery	Restoring Fault Tolerance After the Production Database Has Opened Resetlogs	Restoring Fault Tolerance After the Production Database Has Opened Resetlogs

Recovery Steps for Scheduled Outages

Scheduled outages are planned outages. They are required for regular maintenance of the technology infrastructure that supports the application and include tasks such as hardware maintenance, repair, and upgrades; software upgrades and patching; application changes and patching; and changes to improve performance and manageability of systems. Scheduled outages should be scheduled at times best suited for continual application availability.

Table 9–4 describes the scheduled outages that impact either the primary or secondary site components.

Table 9–4 Scheduled Outages

Outage Class	Description	Examples
Site-wide	The entire site where the current production database resides is unavailable. This is usually known well in advance and can be scheduled.	▪ Scheduled power outages ▪ Site maintenance ▪ Regular planned switchovers to test infrastructure
Hardware maintenance (node impact)	This is scheduled downtime of a database server node for hardware maintenance. The scope of this downtime is restricted to a node of the database cluster.	▪ Repair of a failed component such as a memory card or CPU board ▪ Addition of memory or CPU to an existing node in the database tier
Hardware maintenance (clusterwide impact)	This is scheduled downtime of the database server cluster for hardware maintenance.	▪ Some cases of adding a node to the cluster ▪ Upgrade or repair of the cluster interconnect ▪ Upgrade to the storage tier that requires downtime on the database tier
System software maintenance (node impact)	This is scheduled downtime of a database server node for system software maintenance. The scope of the downtime is restricted to a node.	▪ Upgrade of a software component such as the operating system ▪ Changes to the configuration parameters for the operating system
System software maintenance (clusterwide impact)	This is scheduled downtime of the database server cluster for system software maintenance.	▪ Upgrade or patching of the cluster software ▪ Upgrade of the volume management software

Table 9–4 Scheduled Outages (Cont.)

Outage Class	Description	Examples
Oracle patch upgrade for the database	Scheduled downtime for an Oracle patch	Patch Oracle software to fix a specific customer issue
Oracle patch set or software upgrade for the database	Scheduled downtime for Oracle patch set or software upgrade	■ Patching Oracle software with a patch set ■ Upgrade Oracle software
Database object reorganization	These are changes to the logical structure or the physical organization of Oracle database objects. The primary reason for these changes is to improve performance or manageability. This is always a planned activity. The method and the time chosen to do the reorganization should be planned and appropriate. Using Oracle's online reorganization features enables objects to be available during the reorganization.	■ Moving an object to a different tablespace ■ Converting a table to a partitioned table ■ Renaming or dropping columns of a table

The rest of this section provides outage decision trees for scheduled outages. They appear in the following sections:

- Recovery Steps for Scheduled Outages on the Primary Site
- Recovery Steps for Scheduled Outages on the Secondary Site

The high-level recovery steps for each outage are listed with links to the detailed descriptions for each recovery step. The detailed descriptions of the recovery operations are found in Chapter 10, "Detailed Recovery Steps".

This section also includes the following topic:

- Preparing for Scheduled Secondary Site Maintenance

Recovery Steps for Scheduled Outages on the Primary Site

If the primary site contains the production database and the secondary site contains the standby database, then the outages on the primary site are the ones of most interest. Solutions for these outages are critical for continued availability of the system.

Table 9–5 shows the recovery steps for scheduled outages on the primary site.

Table 9–5 Recovery Steps for Scheduled Outages on the Primary Site

Scope of Outage	Reason for Outage	Recovery Steps for "Database Only" Architecture	Recovery Steps for "RAC Only" Architecture	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Site	Site shutdown	Downtime for entire duration	Downtime for entire duration	1. Database Switchover 2. Complete or Partial Site Failover	1. Database Switchover 2. Complete or Partial Site Failover
Primary database	Hardware maintenance (node impact)	Downtime for entire duration	Managed automatically by RAC Recovery	1. Database Switchover 2. Complete or Partial Site Failover	Managed automatically by RAC Recovery
Primary database	Hardware maintenance (clusterwide impact)	Downtime for entire duration	Downtime for entire duration	1. Database Switchover 2. Complete or Partial Site Failover	1. Database Switchover 2. Complete or Partial Site Failover
Primary database	System software maintenance (node impact)	Downtime for entire duration	Managed automatically by RAC Recovery	1. Database Switchover 2. Complete or Partial Site Failover	Managed automatically by RAC Recovery
Primary database	System software maintenance (clusterwide impact)	Downtime for entire duration	Downtime for entire duration	1. Database Switchover 2. Complete or Partial Site Failover	1. Database Switchover 2. Complete or Partial Site Failover
Primary database	Oracle patch upgrade for the database	Downtime for entire duration	RAC Rolling Upgrade	Downtime for entire duration	RAC Rolling Upgrade
Primary database	Oracle patch set or software upgrade for the database	Downtime for entire duration	Downtime for entire duration	Upgrade with Logical Standby Database	Upgrade with Logical Standby Database

Table 9–5 Recovery Steps for Scheduled Outages on the Primary Site (Cont.)

Scope of Outage	Reason for Outage	Recovery Steps for "Database Only" Architecture	Recovery Steps for "RAC Only" Architecture	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Primary database	Database object reorganization	Online Object Reorganization	Online Object Reorganization	Online Object Reorganization	Online Object Reorganization

Recovery Steps for Scheduled Outages on the Secondary Site

Outages on the secondary site do not impact availability because the clients always access the primary site unless there is a switchover or failover. Outages on the secondary site may affect the MTTR if there are concurrent failures on the primary site. Outages on the secondary site can be managed with no impact on availability. If maximum protection database mode is configured, then downgrade the protection mode before a scheduled outage on the standby instance or database so that there will be no downtime on the production database.

Table 9–6 describes the recovery steps for scheduled outages on the secondary site.

Table 9–6 Recovery Steps for Scheduled Outages on the Secondary Site

Scope of Outage	Reason for Outage	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Site	Site shutdown	Before the outage: "Preparing for Scheduled Secondary Site Maintenance" on page 9-12 After the outage: "Restoring Fault Tolerance after Secondary Site or Clusterwide Scheduled Outage" on page 11-14	Before the outage: "Preparing for Scheduled Secondary Site Maintenance" on page 9-12 After the outage: "Restoring Fault Tolerance after Secondary Site or Clusterwide Scheduled Outage" on page 11-14
Standby database	Hardware or software maintenance the node that is running the managed recovery process (MRP)	Before the outage: "Preparing for Scheduled Secondary Site Maintenance" on page 9-12	Before the outage: "Preparing for Scheduled Secondary Site Maintenance" on page 9-12

Table 9–6 Recovery Steps for Scheduled Outages on the Secondary Site (Cont.)

Scope of Outage	Reason for Outage	Recovery Steps for "Data Guard Only" Architecture	Recovery Steps for MAA
Standby database	Hardware or software maintenance on a node that is not running the MRP	N / A	No impact because the primary standby node or instance receives redo logs that are applied with the managed recovery process After the outage: Restart node and instance when available.
Standby database	Hardware or software maintenance (clusterwide impact)	N / A	Before the outage: "Preparing for Scheduled Secondary Site Maintenance" on page 9-12 After the outage: "Restoring Fault Tolerance after Secondary Site or Clusterwide Scheduled Outage" on page 11-14
Standby database	Oracle patch and software upgrades	Downtime needed for upgrade, but there is no impact on primary node unless the configuration is in maximum protection database mode.	Downtime needed for upgrade, but there is no impact on primary node unless the configuration is in maximum protection database mode.

Preparing for Scheduled Secondary Site Maintenance

To achieve continued service during a secondary site scheduled outage, downgrade the maximum protection mode to maximum availability or maximum performance. When you are scheduling secondary site maintenance, consider that the duration of a site-wide or clusterwide outage adds to the time the standby lags behind the production database, which lengthens the time to restore fault tolerance.

Table 9–7 shows how to prepare for scheduled secondary site maintenance.

Table 9–7 Preparing for Scheduled Secondary Site Maintenance

Production Database Protection Mode	Reason for Outage	Preparation Steps for "Data Guard Only" Architecture and MAA
Maximum protection	Site shutdown	Switch the production data protection mode to either maximum availability or maximum performance **See Also:** "Changing the Data Protection Mode" on page 7-24
Maximum protection	Hardware maintenance (clusterwide impact)	Switch the production data protection mode to either maximum availability or maximum performance **See Also:** "Changing the Data Protection Mode" on page 7-24
Maximum protection	Software maintenance (clusterwide impact)	Switch the production data protection mode to either maximum availability or maximum performance **See Also:** "Changing the Data Protection Mode" on page 7-24
Maximum protection	Hardware maintenance on the primary node (the node that is running the recovery process)	Apply Instance Failover (MAA only) *or* Switch the production data protection mode to either maximum availability or maximum performance
Maximum protection	Software maintenance on the primary node (the node that is running the recovery process)	Apply Instance Failover (MAA only) *or* Switch the production data protection mode to either maximum availability or maximum performance
Maximum availability or maximum performance	Site shutdown	None; no impact on production database

Table 9–7 Preparing for Scheduled Secondary Site Maintenance (Cont.)

Production Database Protection Mode	Reason for Outage	Preparation Steps for "Data Guard Only" Architecture and MAA
Maximum availability or maximum performance	Hardware maintenance (clusterwide impact)	None; no impact on production database
Maximum availability or maximum performance	Software maintenance (clusterwide impact)	None; no impact on production database
Maximum availability or maximum performance	Hardware maintenance on the primary node (the node that is running the recovery process)	Apply Instance Failover (MAA only) *or* None; no impact on production database
Maximum availability or maximum performance	Software maintenance on the primary node (the node that is running the recovery process)	Apply Instance Failover (MAA only) *or* None; no impact on production database

10

Detailed Recovery Steps

This chapter describes the detailed recovery operations that are referred to in the outages and solutions tables in Chapter 9, "Recovering from Outages". It includes the following topics:

- Summary of Recovery Operations
- Complete or Partial Site Failover
- Database Failover
- Database Switchover
- RAC Recovery
- Apply Instance Failover
- Recovery Solutions for Data Failures
- Recovering from User Error with Flashback Technology
- RAC Rolling Upgrade
- Upgrade with Logical Standby Database
- Online Object Reorganization

Summary of Recovery Operations

This chapter describes the detailed recovery operations that are referred to in the outages and solutions tables in Chapter 9, "Recovering from Outages". Table 10–1 summarizes the recovery operations that are described in this chapter.

Table 10–1 Recovery Operations

Recovery Operation	Description
Complete or Partial Site Failover	In the complete site failover scenario, existing connections fail and new connections are routed to a secondary or failover site. This occurs when there is a true disaster and where the application stack is replicated. In the partial site failover scenario, the primary site is intact, and the middle-tier applications need to be redirected after the database has failed over or switched over to a standby database on the secondary site. This configuration is not recommended if performance decreases dramatically because of the greater latency between the application servers and the database.
Database Failover	A Data Guard failover is invoked in the database tier because of an unscheduled outage. If complete recovery is attempted, then there is minimal or no data loss. A subsequent complete or partial site failover must occur. The previous production database can be converted to a standby database by flashing back the database.
Database Switchover	A Data Guard switchover occurs in the database tier. The previous production database becomes the new standby database while the previous standby database becomes the new production database. A subsequent complete or partial site failover must occur. This is a scheduled or planned outage.
RAC Recovery	RAC automatically handles instance and node failures at a given site to provide continued access to the backend database. The reconnection to available instances can be transparent from an application standpoint and occurs only on the primary site. The application service migrates to the available or designated instances.
Apply Instance Failover	When a standby node requires maintenance, you can switch to another standby instance to avoid any impact on the production database and to ensure that the standby recovery does not fall behind. When the standby cluster requires maintenance or the standby cluster fails, if the maximum protection mode is enabled, then the production database needs to be downgraded to maximum availability mode or maximum performance mode.
Application failover	The client or application tier automatically fails over to one or more surviving RAC instances when an instance or node failure occurs at the primary site. **See Also**: "Recommendations for Fast Application Failover" on page 7-44
Recovery Solutions for Data Failures	When data failure occurs due to media corruption or media damage, you can use different recovery options that use the flash recovery area or switch over to the standby database. Alternatively, you can rebuild tables by fast imports or rebuild indexes in parallel.

Table 10–1 Recovery Operations (Cont.)

Recovery Operation	Description
Recovering from User Error with Flashback Technology	When a user error causes transactional or logical data inconsistencies, you can resolve these problems by using flashback error correction technology at the row, transaction, table, or database levels.
RAC Rolling Upgrade	With RAC, you can apply patches for specific customer issues incrementally to one node or instance at a time, which enables continual application and database availability.
Upgrade with Logical Standby Database	Data Guard enables you to upgrade the database on the standby database and perform a switchover, which minimizes the overall scheduled outage time for applying patch sets or software upgrades for the database.
Online Object Reorganization	Many scheduled outages related to the data server involve some reorganization of the database objects. They need to be accomplished with continued availability of the database. Oracle online object reorganization is used to manage the scheduled outages.

Complete or Partial Site Failover

This section describes the following client failover scenarios:

- Complete Site Failover
- Partial Site Failover: Middle-Tier Applications Connect to a Remote Database Server

In the complete site failover scenario, existing connections fail and new connections are routed to a secondary or failover site. This occurs when there is a true disaster and where the application stack is replicated.

In the partial site failover scenario, the primary site is intact, and the middle-tier applications need to be redirected after the database has failed over or switched over to a standby database on the secondary site. This configuration is not recommended if performance decreases significantly because of the greater latency between the application servers and the database.

Complete Site Failover

A wide-area traffic manager is implemented on the primary and secondary sites to provide the site failover function. The wide-area traffic manager can redirect traffic automatically if the primary site or a specific application on the primary site is not accessible. It can also be triggered manually to switch to the secondary site for switchovers. Traffic is directed to the secondary site only when the primary site

cannot provide service due to an outage or after a switchover. If the primary site fails, then user traffic is directed to the secondary site.

Figure 10–1 illustrates the network routes before site failover. Client requests enter the client tier of the primary site and travel by the WAN traffic manager. Client requests are sent through the firewall into the demilitarized zone (DMZ) to the application server tier. Requests are then forwarded through the active load balancer to the application servers. They are then sent through another firewall and into the database server tier. The application requests, if required, are routed to a RAC instance. Responses are sent back to the application and clients by a similar path.

Figure 10–1 Network Routes Before Site Failover

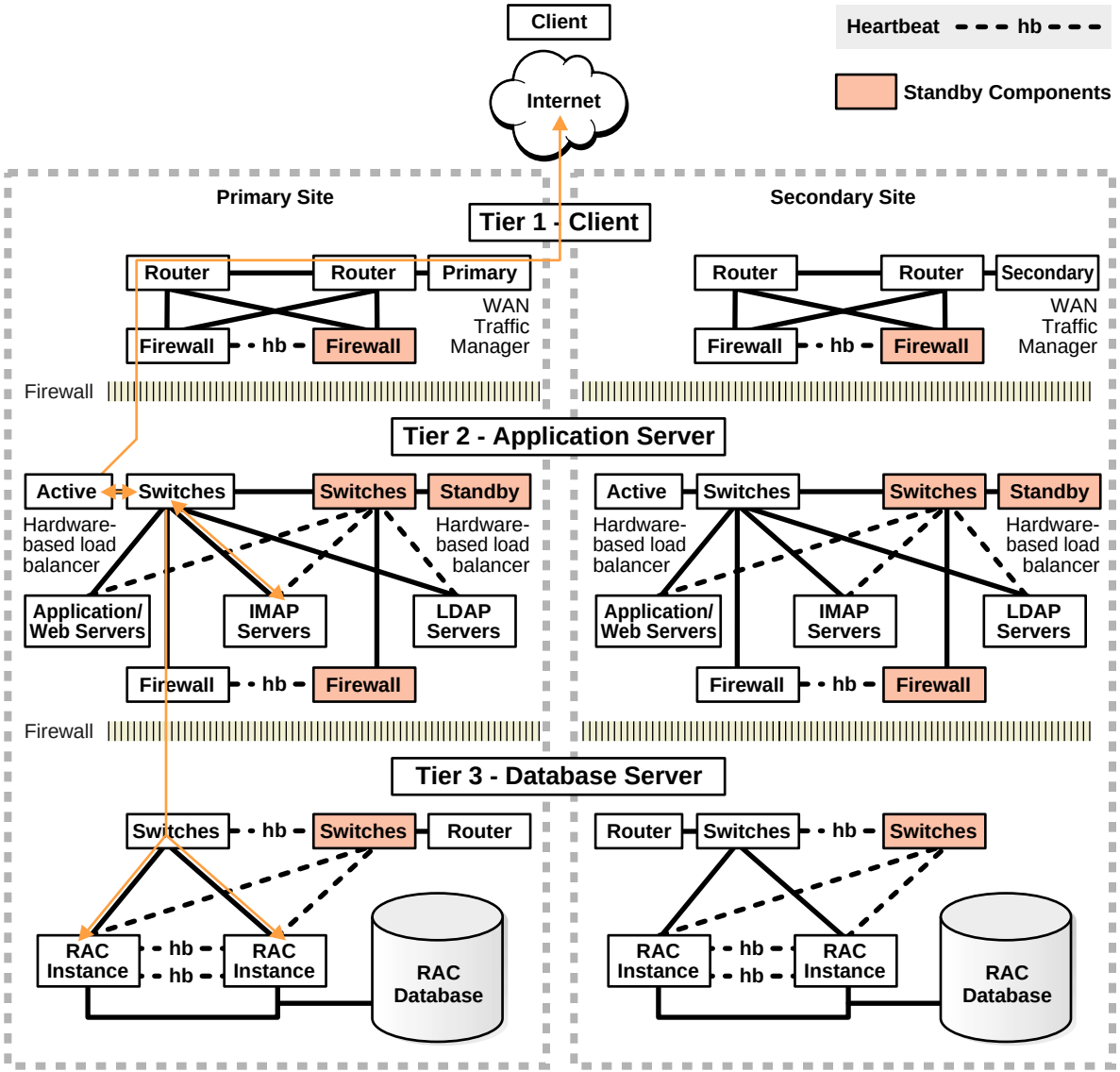
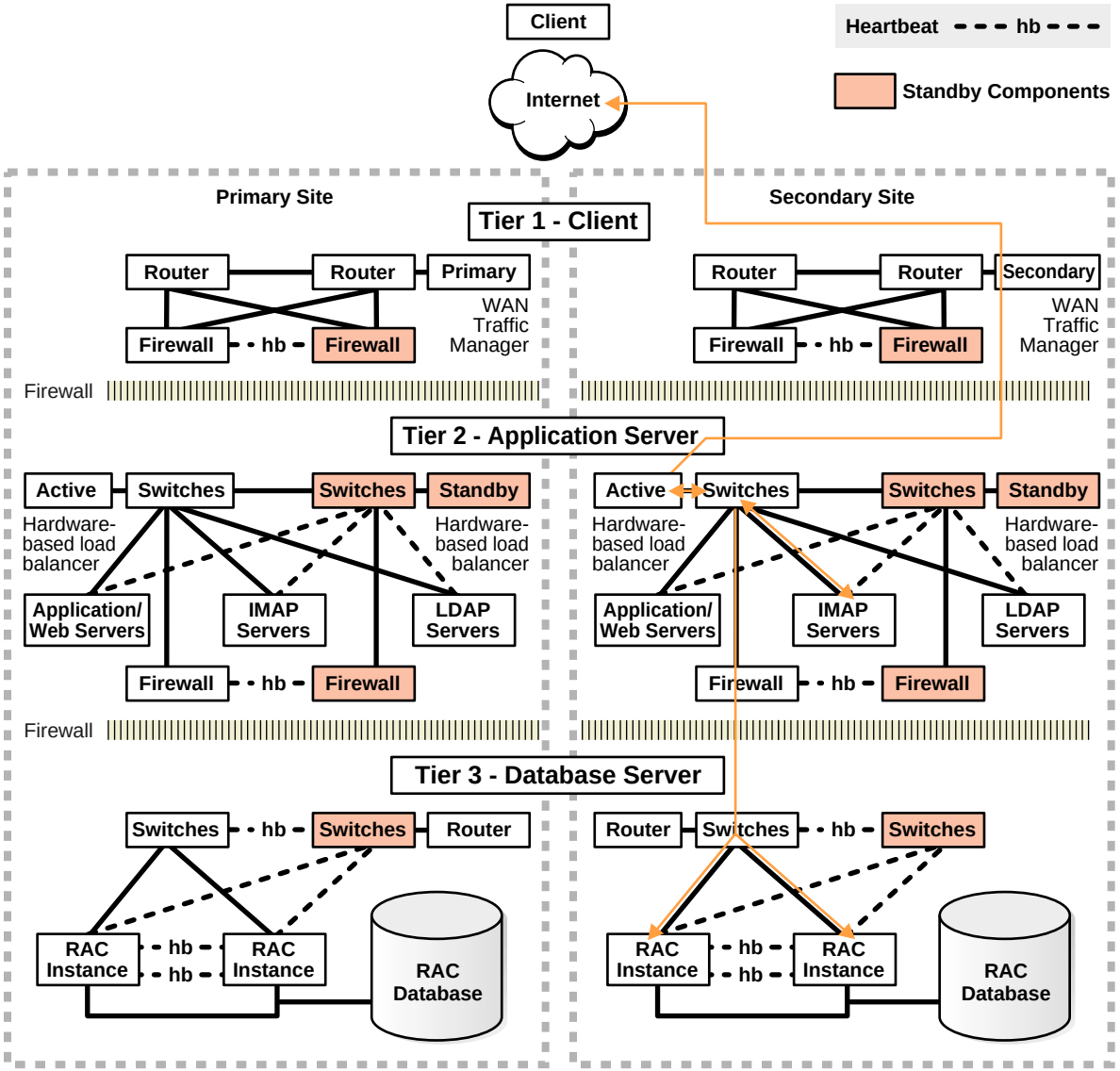


Figure 10–2 illustrates the network routes after site failover. Client or application requests enter the secondary site at the client tier and follow exactly the same path on the secondary site that they followed on the primary site.

Figure 10–2 Network Routes After Site Failover



The following steps describe what happens to network traffic during a failover or switchover.

1. The administrator has failed over or switched over the production database to the secondary site.
2. The administrator starts the middle-tier application servers on the secondary site.
3. Typically, a Domain Name Service (DNS) administrator changes the wide-area traffic manager selection of the secondary site. Alternatively, the selection can be made automatically for an entire site failure.The wide-area traffic manager at the secondary site returns the virtual IP address of a load balancer at the secondary site. In this scenario, the site failover is accomplished by a DNS failover. The following is an example of a manual DNS failover:
   a. Change the DNS to point to the secondary site load balancer.
   b. Set TTL (Time to Live) to a short interval for the DNS propagation.
   c. Disable DNS on the primary site.
   d. Execute a DNS "push" to point to the secondary site.
   e. Wait until all failover operations are complete.
   f. Change TTL back to its normal setting on the DNS server.
   g. The secondary site load balancer directs traffic to the secondary site middle-tier application server.
   h. The secondary site is ready to take client requests.

Failover also depends on the client's web browser. Most browser applications cache the DNS entry for a period of time. Consequently, sessions in progress during an outage may not fail over until the cache timeout expires. The only way to resume service to such clients is to close the browser and restart it.

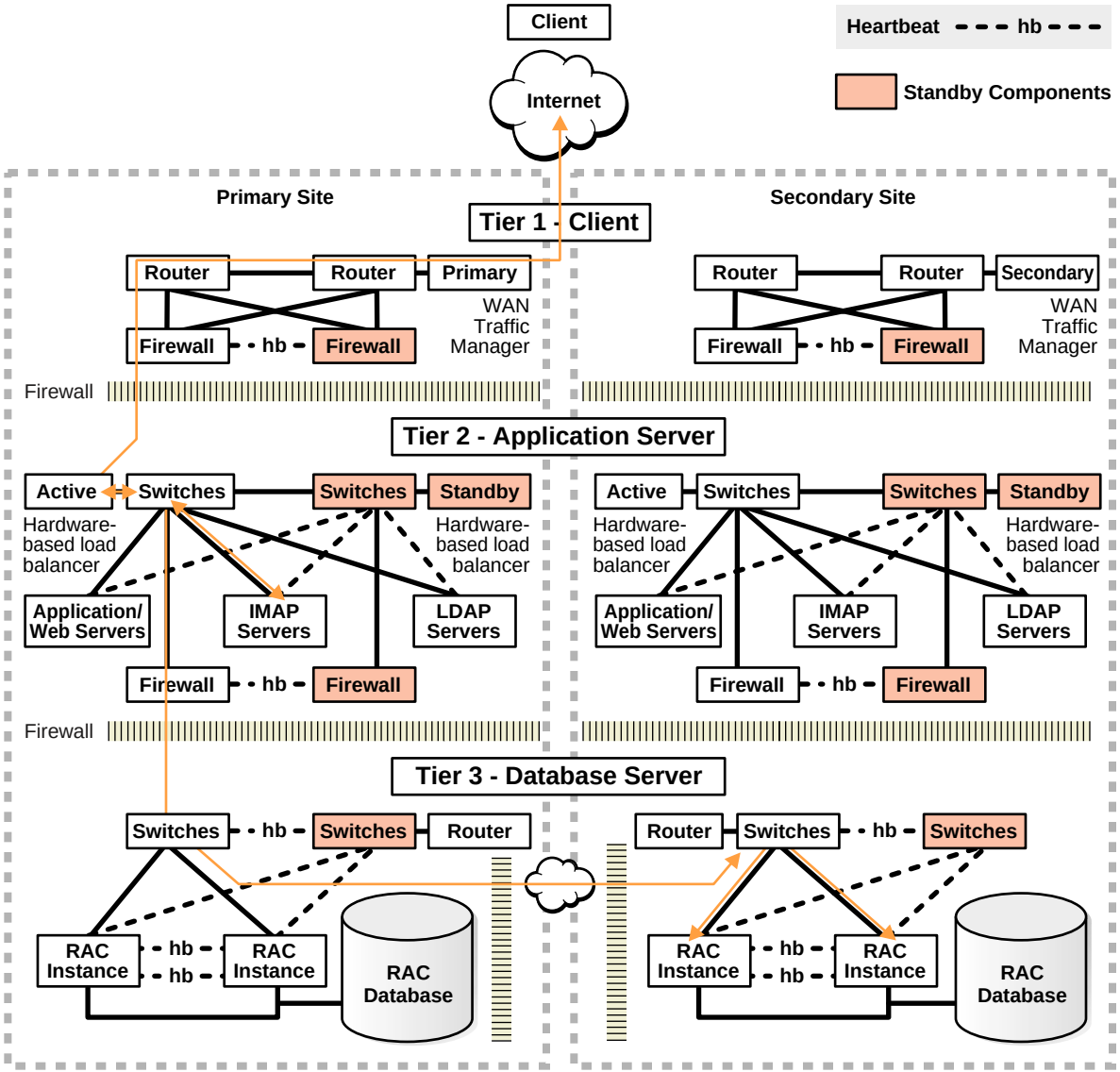
Partial Site Failover: Middle-Tier Applications Connect to a Remote Database Server

This usually occurs after the database has been failed over or switched over to the secondary site and the middle-tier applications remain on the primary site. The following steps describe what happens to network traffic during a partial site failover:

1. The production database is failed over or switched over to the secondary site.
2. The middle-tier application servers reconnect to the database on the secondary site using configuration best practices described in "Recommendations for Fast Application Failover" on page 7-44.

Figure 10–3 shows the network routes after partial site failover. Client and application requests enter the primary site at the client tier and follow the same path to the database server tier as in Figure 10–1. When the requests enter the database server tier, they are routed to the database tier of the secondary site through any additional switches, routers, and possible firewalls.

Figure 10–3 Network Routes After Partial Site Failover



Database Failover

Failover is the operation of taking the production database offline on one site and bringing one of the standby databases online as the new production database. A failover operation can be invoked when an unplanned catastrophic failure occurs on the production database, and there is no possibility of recovering the production database in a timely manner.

Data Guard enables you to fail over by issuing the SQL statements described in subsequent sections, by using Oracle Enterprise Manager, or by using the Oracle Data Guard broker command-line interface.

> **See Also:** *Oracle Data Guard Broker* for information about using Enterprise Manager or the Data Guard broker command-line for database failover

Data Guard failover is a series of steps to convert a standby database into a production database. The standby database essentially assumes the role of production. A Data Guard failover is accompanied by a site failover to fail over the users to the new site and database. After the failover, the secondary site contains the production database. The former production database needs to be re-created as a new standby database to restore resiliency. The standby database can be quickly re-created by using Flashback Database. See "Restoring the Standby Database After a Failover" on page 11-10.

During a failover operation, little or no data loss may be experienced. The complete description of a failover can be found in *Oracle Data Guard Concepts and Administration*.

The rest of this section includes the following topics:

- When to Use Data Guard Failover
- When Not to Use Data Guard Failover
- Data Guard Failover Using SQL*Plus

When to Use Data Guard Failover

Data Guard failover should be used only in the case of an emergency and should be initiated due to an unplanned outage such as:

- Site disaster
- User errors

- Data failures

A failover requires that the initial production database be re-created as a standby database to restore fault tolerance to your environment. The standby database can be quickly re-created by using Flashback Database. See "Restoring the Standby Database After a Failover" on page 11-10.

When Not to Use Data Guard Failover

Do not use Data Guard failover when the problem can be fixed locally in a timely manner or when Data Guard switchover can be used. For failover with complete recovery scenarios, either the production database is not accessible or cannot be restarted. Data Guard failover should not be used where object recovery or flashback technology solutions provide a faster and more efficient alternative.

Data Guard Failover Using SQL*Plus

This section includes the following topics:

- Physical Standby Failover Using SQL*Plus
- Logical Standby Failover Using SQL*Plus

Physical Standby Failover Using SQL*Plus

1. Check for archive gaps.

```
SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE#
FROM V$ARCHIVE_GAP;
```

> **See Also:** *Oracle Data Guard Concepts and Administration* for more information about what to do if a gap exists

2. Shut down other standby instances.
3. Finish recovery.

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
```

4. Check the database state. Query switchover status readiness by executing the following statement:

```
SELECT SWITCHOVER_STATUS FROM V$DATABASE;
```

5. Convert the physical standby database to the production role.

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

6. Restart instance.

Logical Standby Failover Using SQL*Plus

1. Stop the current SQL Apply session.

   ```
   ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   ```

2. If there are additional logs to be registered, (for example, you can get to the primary database or you are using LGWR to the standby destination), then register the log files.

   ```
   ALTER DATABASE REGISTER LOGICAL LOGFILE 'file_name';
   ```

3. Start the SQL Apply session using the NODELAY and FINISH clauses.

   ```
   ALTER DATABASE START LOGICAL STANDBY APPLY NODELAY FINISH;
   ```

4. Activate the logical standby database.

   ```
   ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE;
   ```

Failover has completed, and the new production database is available to process transactions.

Database Switchover

A database switchover performed by Oracle Data Guard is a planned transition that includes a series of steps to switch roles between a standby database and a production database. Thus, following a successful switchover operation, the standby database assumes the production role and the production database becomes a standby database. In a RAC environment, a switchover requires that only one instance is active for each database, production and standby. At times the term "switchback" is also used within the scope of database role management. A switchback operation is a subsequent switchover operation to return the roles to their original state.

Data Guard enables you to change these roles dynamically by issuing the SQL statements described in subsequent sections, or by using Oracle Enterprise Manager, or by using the Oracle Data Guard broker command-line interface. Using Oracle Enterprise Manager or the Oracle Data Guard broker command-line interface is described in *Oracle Data Guard Broker*.

This section includes the following topics:

- When to Use Data Guard Switchover
- When Not to Use Data Guard Switchover
- Data Guard Switchover Using SQL*Plus

When to Use Data Guard Switchover

Switchover is a planned operation. Switchover is the capability to switch database roles between the production and standby databases without needing to instantiate any of the databases. Switchover can occur whenever a production database is started, the target standby database is available, and all the archived redo logs are available. It is useful in the following situations:

- Scheduled maintenance such as hardware maintenance (such as hardware or firmware patches) on the production host
- Resolution of data failures when the production database is still opened
- Testing and validating the secondary resources, as a means to test disaster recovery readiness
- Upgrades for database software and patch sets. See "Upgrade with Logical Standby Database" on page 10-49.

When Not to Use Data Guard Switchover

Switchover is not possible or practical under the following circumstances:

- Missing archived redo logs
- Point-in-time recovery is required
- The production database is not open and cannot be opened

Do not use Data Guard switchover when local recovery solutions provide a faster and more efficient alternative. The complete description of a switchover can be found in *Oracle Data Guard Concepts and Administration*.

Data Guard Switchover Using SQL*Plus

If you are not using Oracle Enterprise Manager, then the high-level steps in this section can be executed with SQL*Plus. These steps are described in detail in *Oracle Data Guard Concepts and Administration*.

This section includes the following topics:

- Physical Standby Switchover Using SQL*Plus
- Logical Standby Switchover Using SQL*Plus

Physical Standby Switchover Using SQL*Plus

1. Shut down all production and standby instances except one for each site.
2. Stop active sessions on remaining active production instance

   To identify active sessions, execute the following query:

   ```
   SELECT SID, PROCESS, PROGRAM
   FROM V$SESSION
   WHERE TYPE = 'USER'
   AND SID <> (SELECT DISTINCT SID FROM V$MYSTAT);
   ```

3. Check that the switchover status on the production database is 'TO STANDBY'.

   ```
   SELECT SWITCHOVER_STATUS FROM V$DATABASE;
   ```

4. Switch over the current production database to the standby database.

   ```
   ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY [WITH SESSION SHUTDOWN];
   ```

5. Start the new standby database.

   ```
   STARTUP MOUNT;
   RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT;
   ```

6. Convert the former standby database to a production database.

   ```
   ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY [WITH SESSION SHUTDOWN];
   ```

7. Restart all instances.

Logical Standby Switchover Using SQL*Plus

1. Prepare the production database to become the logical standby database.

   ```
   ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY;
   ```

2. Prepare the logical standby database to become the production database.

   Following this step, logs start to ship in both directions, although the current production database does not process the logs coming from the current logical standby database.

```
ALTER DATABASE PREPARE TO SWITCHOVER TO PRIMARY;
```

3. Commit the production database to become the logical standby database.

   This is the phase where current transactions on the production database are cancelled. All DML-related cursors are invalidated, preventing new records from being applied. The end of redo (EOR) marker is recorded in the online redo log and then shipped (immediately if using real-time apply) to the logical standby database and registered.

   ```
   ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;
   ```

4. Commit the logical standby database to become the production database.

   ```
   ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
   ```

5. Start the logical standby apply engine on the new logical standby database.

   If real-time apply is required, execute the following statement:

   ```
   ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
   ```

   Otherwise execute the following statement:

   ```
   ALTER DATABASE START LOGICAL STANDBY APPLY;
   ```

RAC Recovery

This section includes the following topics:

- RAC Recovery for Unscheduled Outages
- RAC Recovery for Scheduled Outages

RAC Recovery for Unscheduled Outages

This section includes the following topics:

- Automatic Instance Recovery for Failed Instances
- Automatic Service Relocation

Automatic Instance Recovery for Failed Instances

Instance failure occurs when software or hardware problems disable an instance. After instance failure, Oracle automatically uses the online redo log file to perform database recovery as described in this section.

Single Node Failure in Real Application Clusters Instance recovery in RAC does not include restarting the failed instance or the recovery of applications that were running on the failed instance. Applications that were running continue by using failure recognition and recovery as described in *Oracle Real Application Clusters Installation and Configuration Guide*. This provides consistent and uninterrupted service in the event of hardware or software failures. When one instance performs recovery for another instance, the surviving instance reads redo log entries generated by the failed instance and uses that information to ensure that committed transactions are recorded in the database. Thus, data from committed transactions is not lost. The instance that is performing recovery rolls back uncommitted transactions that were active at the time of the failure and releases resources used by those transactions.

Multiple Node Failures in Real Application Clusters When multiple node failures occur, as long as one instance survives, RAC performs instance recovery for any other instances that fail. If all instances of a RAC database fail, then Oracle automatically recovers the instances the next time one instance opens the database. The instance that is performing recovery can mount the database in either shared or exclusive mode from any node of a RAC database. This recovery procedure is the same for Oracle running in shared mode as it is for Oracle running in exclusive mode, except that one instance performs instance recovery for all the failed instances in exclusive node.

Automatic Service Relocation

Service reliability is achieved by configuring and failing over among redundant instances. More instances are enabled to provide a service than would otherwise be needed. If a hardware failure occurs and adversely affects a RAC database instance, then RAC automatically moves any services on that instance to another available instance. Then Cluster Ready Services (CRS) attempts to restart the failed nodes and instances.

An installation can specify the "preferred" and "available" configuration for each service. This configuration describes the preferred way to run the system, and is used when the service first starts up. For example, the ERP service runs on `instance1` and `instance2`, and the HR service runs on `instance3` when the system first starts. `instance2` is available to run HR in the event of a failure or planned outage, and `instance3` and `instance4` are available to run ERP. The service configuration can be designed several ways.

RAC recognizes when a failure affects a service and automatically fails over the service and redistributes the clients across the surviving instance supporting the service. In parallel, CRS attempts to restart and integrate the failed instances and

dependent resources back into the system. Notification of failures occur at various levels including notifying external parties through Enterprise Manager and callouts, recording the fault for tracking, event logging, and interrupting applications. Notification occurs from a surviving fault domain when the failed domain is out of service. The location and number of fault domains serving a service is transparent to the applications. Auto restart and recovery are automatic, including all the subsystems, not just database.

RAC Recovery for Scheduled Outages

This section includes the following topics:

- Disabling CRS-Managed Resources
- Planned Service Relocation

Disabling CRS-Managed Resources

When an outage occurs, RAC automatically restarts essential components. Components that are eligible for automatic restart include instances, listeners, and the database as well as several subcomponents. Some scheduled administrative tasks require that you prevent components from automatically restarting. To perform scheduled maintenance that requires a CRS-managed component to be down during the operation, the resource must be disabled to prevent CRS from trying to automatically restart the component. For example, to take a node and all of its instances and services offline for maintenance purposes, disable the instance and its services using either Enterprise Manager or SRVCTL, and then perform the required maintenance. Otherwise, if the node fails and then restarts, then CRS attempts to restart the instance during the administrative operation.

Planned Service Relocation

For a scheduled outage that requires an instance, node, or other component to be isolated, RAC provides the ability to relocate, disable, and enable services. Relocation migrates the service to another instance. The sessions can also be relocated. These interfaces also allow services, instances and databases to be selectively disabled while a repair, change, or upgrade is made and re-enabled after the change is complete. This ensures that the service is not started at the instance being repaired because of a dependency or a start operation on the service. The service is disabled on the instance at the beginning of the planned outage. It is then enabled at the end of the maintenance outage.

For example, to relocate the SALES service from instance1 to instance3 in order to perform scheduled maintenance on node1,the tasks can be performed using Enterprise Manager or SRVCTL commands. The following shows how to use SRVCTL commands:

1. Relocate the SALES service to instance3.

```
srvctl relocate service -d PROD -s SALES -i instance1 -t instance3
```

2. Disable the SALES service on instance1 to prevent it from being relocated to instance1 while maintenance is performed.

```
srvctl disable service -d PROD -s SALES -i instance1
```

3. Stop instance1.

```
srvctl stop instance -d PROD -i instance1
```

4. Perform the scheduled maintenance.
5. Start instance1.

```
srvctl start instance -D PROD -i instance1
```

6. Re-enable the SALES service on instance1.

```
srvctl enable service -d PROD -s SALES -i instance1
```

7. If desired, relocate the SALES service running on instance3 back to instance1.

```
srvctl relocate service -d PROD -s SALES -i instance3 -t instance1
```

Note Also: *Oracle Real Application Clusters Administrator's Guide*

Apply Instance Failover

This section applies to MAA, with RAC and Data Guard on each site.

A standby database can have multiple standby instances. Only one instance can have the managed recovery process (MRP) or the logical standby apply process (LSP). The instance with the MRP or LSP is called the apply instance.

When you have a RAC-enabled standby database, you can fail over the apply instance of the standby RAC environment. Failing over to another apply instance may be necessary when incurring a planned or unplanned outage that affects the apply instance or node. Note the difference between apply instance failover, which

utilizes multiple instances of the standby database at the secondary site, and Data Guard failover or Data Guard switchover, which converts the standby database into a production database. The following occurs as a result of apply instance failover:

- Standby recovery can continue even when the apply host is undergoing maintenance or incurs a failure. By not interrupting standby recovery, you can ensure that a Data Guard failover or switchover can be completed within the tolerated MTTR.
- The production database will not be interrupted and production downtime can be avoided even if you are using maximum protection mode, because the subsequent network connection switches to another apply instance.

For apply failover to work correctly, "Configuration Best Practices for MAA" on page 7-35 must be followed:

- Configure Multiple Standby Instances
- Configure Connect-Time Failover for Network Service Descriptors

When you follow these configuration recommendations, apply instance failover is automatic for a scheduled or unscheduled outage on the primary instance, and all standby instances have access to archived redo logs. By definition, all RAC standby instances already have access to standby redo logs because they must reside on shared storage.

The method of restarting the physical standby managed recovery process (MRP) or the logical standby apply process (LSP) depends on whether Data Guard Broker is being used. If the Data Guard Broker is in use, then the MRP or LSP is automatically restarted on the first available standby instance if the primary standby instance fails. If the Data Guard Broker is not being used, then the MRP or LSP must be manually restarted on the new standby instance. Consider using a shared file system, such as a clustered file system or a global file system, for the archived redo logs. A shared file system enables you to avoid reshipment of any unapplied archived redo logs that were already shipped to the standby.

> **See Also:** *Oracle Data Guard Concepts and Administration* for details about setting up cross-instance archiving

Performing an Apply Instance Failover Using SQL*Plus

If apply instance failover does not happen automatically, then follow these steps to restart your production database, if necessary, and restart MRP or LSP following an unscheduled apply instance or node outage:

- Step 1: Ensure That the Chosen Standby Instance is Mounted
- Step 2: Verify Oracle Net Connection to the Chosen Standby Host
- Step 3: Start Recovery on the Chosen Standby Instance
- Step 4: Copy Archived Redo Logs to the New Apply Host
- Step 5: Verify the New Configuration

Step 1: Ensure That the Chosen Standby Instance is Mounted

From the targeted standby instance, run the following query.

```
SELECT OPEN_MODE, DATABASE_ROLE FROM V$DATABASE;
```

Type of Standby Database	Output for Mounted Standby Database	If Not Mounted, Choose a Different Target or Open Manually
Physical	`MOUNTED, PHYSICAL STANDBY`	`STARTUP NOMOUNT`
Logical	`READ WRITE, LOGICAL STANDBY`	`STARTUP`

Step 2: Verify Oracle Net Connection to the Chosen Standby Host

1. Ensure that the standby listener is started.

   ```
   % lsnrctl status listener_name
   ```

2. Validate the Oracle Net alias from all the production hosts in the cluster.

   ```
   % tnsping standby_database_connection_service_name
   ```

If the connection cannot be made, then consult *Oracle Net Services Administrator's Guide* for further troubleshooting.

Step 3: Start Recovery on the Chosen Standby Instance

Use the following statements for a physical standby database:

```
RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT;
```

Use the following statements for a logical standby database:

```
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

Step 4: Copy Archived Redo Logs to the New Apply Host

Optionally, copy the archived redo logs to the new apply host.

The copy is not necessary for a physical standby database. For a physical standby database, when the managed recovery process detects an archive gap, it requests the production archived redo logs to be resent automatically.

For a logical standby database, unapplied archive file names that have already been registered are not resent automatically to the logical standby database. These archived redo logs must be sent manaully to the same directory structure in the new apply host. You can identify the registered unapplied archived redo logs by executing a statement similar to the following:

```
SELECT LL.FILE_NAME, LL.THREAD#, LL.SEQUENCE#, LL.FIRST_CHANGE#, LL.NEXT_CHANGE#,
LP.APPLIED_SCN, LP.READ_SCN
  FROM DBA_LOGSTDBY_LOG LL, DBA_LOGSTDBY_PROGRESS LP
  WHERE LEAST(LP.APPLIED_SCN, LP.READ_SCN) <= LL.NEXT_CHANGE#;
```

Compare the results of the statement to the contents of the STANDBY_ARCHIVE_DEST directory.

> **See Also:** "Oracle9i Data Guard: SQL Apply Best Practices" at `http://otn.oracle.com/deploy/availability/htdocs/maa.htm`

Step 5: Verify the New Configuration

1. Verify that archived redo logs are being sent to the new apply host.

   Query V$ARCHIVE_STATUS and V$ARCHIVED_DEST_STATUS.

   ```
   SELECT NAME_SPACE, STATUS, TARGET, LOG_SEQUENCE ,
   TYPE,PROCESS, REGISTER , ERROR FROM V$ARCHIVE_DEST
   WHERE STATUS!='INACTIVE';

   SELECT * FROM V$ARCHIVE_DEST_STATUS WHERE STATUS!='INACTIVE';
   ```

2. Verify that the managed recovery or logical apply on the new apply host is progressing.

   Issue the following queries to ensure that the sequence number is advancing over time.

   Use the following statements for a physical standby database:

   ```
   SELECT MAX(SEQUENCE#), THREAD# FROM V$LOG_HISTORY GROUP BY THREAD#;
   SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, CLIENT_PROCESS FROM V$MANAGED_ STANDBY;
   ```

Use the following statements for a logical standby database:

```
SELECT MAX(SEQUENCE#), THREAD# FROM DBA_LOGSTDBY_LOG GROUP BY THREAD#;
SELECT APPLIED_SCN FROM DBA_LOGSTDBY_PROGRESS;
```

Recovery Solutions for Data Failures

Recovering from a data failure is an unscheduled outage scenario. A data failure is usually, but not always, caused by some activity or failure that occurs outside the database, even though the problem may be evident within the database.

Data failure can affect the following types of database objects:

- Datafiles such as application tablespaces, data dictionary or UNDO tablespace, temporary tablespace
- Control file
- Standby control file
- Online redo log
- Standby redo log
- Archived redo log
- Server parameter file (SPFILE)

Data failure can be categorized as either datafile block corruption or media failure:

- Datafile block corruption: A corrupt datafile block can be accessed but the contents within the block are invalid or inconsistent.
- Media failure: Media failure results from a physical hardware problem or user error. The system cannot successfully read or write to a file that is necessary to operate the database.

In all environments, you can resolve a data failure outage by one of the following methods:

- RMAN datafile media restoration and recovery
- RMAN block media restoration and recovery
- Manually re-create the object

In a Data Guard environment, you can also use a Data Guard switchover or failover to a standby database to recover from data failures.

Another category of related outages that result in database objects becoming unavailable or inconsistent are caused by user error, such as dropping a table or erroneously updating table data. Information about recovering from user error can be found in "Recovering from User Error with Flashback Technology" on page 10-35.

The rest of this section includes the following topics:

- Detecting and Recovering From Datafile Block Corruption
- Recovering From Media Failure
- Recovery Methods for Data Failures

Detecting and Recovering From Datafile Block Corruption

A corrupt datafile block can be accessed, but the contents within the block are invalid or inconsistent. The typical cause of datafile corruption is a faulty hardware or software component in the I/O stack, which includes, but is not limited to, the file system, volume manager, device driver, host bus adapter, storage controller, and disk drive.

The database usually remains available when corrupt blocks have been detected, but some corrupt blocks may cause widespread problems, such as corruption in a file header or with a data dictionary object, or corruption in a critical table that renders an application unusable.

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide* for advanced user-managed scenarios
- "Configuration Best Practices for Data Guard" on page 7-13

The rest of this section includes the following topics:

- Detecting Datafile Block Corruption
- Recovering From Datafile Block Corruption

Detecting Datafile Block Corruption

A data fault is detected when it is recognized by the user, administrator, RMAN backup, or application because it has affected the availability of the application. For example:

- A single corrupt data block in a user table that cannot be read by the application because of a bad spot of the physical disk
- A database that automatically shuts down because of the invalid blocks of a datafile in the SYSTEM tablespace caused by a failing disk controller

Regularly monitor application logs (which may be distributed across the data server, middle-tier and the client machines), the alert log, and Oracle trace files for errors such as ORA-1578 and ORA-1110

```
ORA-01578: ORACLE data block corrupted (file # 4, block # 26)
ORA-01110: data file 4: '/u01/oradata/objrs/obj_corr.dbf'
```

Recovering From Datafile Block Corruption

After you have identified datafile block corruption, follow these steps:

1. Determine the Extent of the Corruption Problem
2. Replace or Move Away From Faulty Hardware
3. Determine Which Objects Are Affected
4. Decide Which Recovery Method to Use

Determine the Extent of the Corruption Problem Use the following methods to determine the extent of the corruption:

- Gather details from error messages

  Gather the file number, file name, and block number from the error messages. For example:

  ```
  ORA-01578: ORACLE data block corrupted (file # 22, block # 12698)
  ORA-01110: data file 22: '/oradata/SALES/users01.dbf'
  ```

  The file number is 22, the block number is 12698, and the file name is `/oradata/SALES/users01.dbf`.

- Check log files for additional error messages

  Record additional error messages that appear in the alert log, Oracle trace files, or application logs. Note that log files may be distributed across the data server, middle tier, and client machines.

- Use Oracle utilities to check for additional corruption

  Use Oracle detection tools to find other data failure problems that may exist on the same disk or set of disks that have not yet been reported. For example, if the

file number 22 has corrupt blocks, then it is prudent to run the RMAN BACKUP VALIDATE DATAFILE 22 command to detect additional corruption. Table 10–2 summarizes the Oracle tools that are available to detect datafile block corruption.

Table 10–2 Oracle Tools for Detecting Datafile Block Corruption

Oracle Tool	Description	Location of Additional Information
RMAN BACKUP or RESTORE command with VALIDATE option	RMAN scans the specified files and verifies content-checking for physical and logical errors but does not actually perform the backup or recovery operation. Oracle records the address of the corrupt block and the type of corruption in the control file. Access these records through the V$DATABASE_BLOCK_CORRUPTION view, which can be used by RMAN block media recovery. If BLOCK CHANGE TRACKING is enabled, then do *not* use the INCREMENTAL LEVEL option with BACKUP VALIDATE to ensure that all data blocks are read and verified. To detect all types of corruption that are possible to detect: ■ Do *not* specify the MAXCORRUPT option ■ Do *not* specify the NOCHECKSUM option ■ *Do* specify the CHECK LOGICAL option	*Oracle Database Backup and Recovery Advanced User's Guide*
DBVERIFY utility	External command-line utility that performs physical structure data integrity check on a datafile or a specific segment within a datafile. Output of DBVERIFY goes to the screen or the specified LOGFILE.	*Oracle Database Utilities*
ANALYZE SQL statement with VALIDATE STRUCTURE option	Verifies the integrity of the structure of an index, table, or cluster; identifies corruption; and checks or verifies that your tables and indexes are consistent. Problems are reported into the user session trace file in USER_DUMP_DEST.	*Oracle Database Administrator's Guide*
DBMS_REPAIR PL/SQL package	Performs block checking for a specified table, partition, or index. DBMS_REPAIR should be used with care and under the guidance of Oracle Support.	*Oracle Database Administrator's Guide*

Replace or Move Away From Faulty Hardware Some corruption problems are caused by faulty hardware. If there is a hardware fault or a suspect component, then it is sensible to either repair the problem, or make disk space available on a separate disk subsystem before proceeding with a recovery option.

If there are multiple errors, if there are operating system-level errors against the affected file, or if the errors are transient and keep moving about, then there is little point in proceeding until the underlying problem has been addressed or space is available on alternative disks. Ask your hardware vendor to verify system integrity.

In a Data Guard environment, a switchover can be performed to bring up the application and restore service quickly while the corruption problem is handled offline.

Determine Which Objects Are Affected Using the file ID (*fid*) and block ID (*bid*) gathered from error messages and the output from Oracle block checking utilities, determine which database objects are affected by the corruption by using a query similar to the following:

```
SELECT tablespace_name, partition_name, segment_type,
       owner, segment_name
FROM dba_extents
WHERE file_id = fid
  AND bid BETWEEN block_id AND block_id + blocks - 1;
```

The following is an example of the query and its resulting output:

```
SQL> select tablespace_name, partition_name, segment_type,
  2  owner, segment_name from dba_extents
  3  where file_id=4 and 11 between block_id and block_id + blocks -1;

TABLESPACE_NAME   PARTITION_NAME     SEGMENT_TYPE    OWNER   SEGMENT_NAME
---------------   --------------     ------------    -----   ------------
USERS                                TABLE           SCOTT   EMP
```

The integrity of a table or index can be determined by using the ANALYZE statement.

Decide Which Recovery Method to Use The recommended recovery methods are summarized in Table 10–3 and Table 10–4. The recovery methods depend on whether Data Guard is being used.

Table 10–3 summarizes recovery methods for data failure when Data Guard is not used.

Table 10–3 Recovering From Data Failure Without Data Guard

Object Affected	Extent of Problem	Action
Data dictionary or UNDO segment	N / A	Use RMAN Datafile Media Recovery
Application segment (user table, index, cluster)	Widespread or unknown	Use RMAN Datafile Media Recovery or Re-Create Objects Manually
Application segment (user table, index, cluster)	Localized	Use RMAN Block Media Recovery or Re-Create Objects Manually
TEMPORARY segment or temporary table	N / A	No impact to permanent objects. Re-create temporary tablespace if required.

Table 10–4 summarizes recovery methods for data failure when Data Guard is present.

Table 10–4 Recovering from Data Failure With Data Guard

Object Affected	Extent of Problem	Impact to Application	Cost of Local Recovery	Action
Data dictionary or UNDO segment	N / A	N / A	N / A	Use Data Guard to Recover From Data Failure
Application segment (user table, index, cluster)	Widespread or unknown	Low	Low	Use RMAN Datafile Media Recovery or Re-Create Objects Manually
Application segment (user table, index, cluster)	N / A	High	N / A	Use Data Guard to Recover From Data Failure
Application segment (user table, index, cluster)	N / A	N / A	High	Use Data Guard to Recover From Data Failure

Table 10–4 Recovering from Data Failure With Data Guard (Cont.)

Object Affected	Extent of Problem	Impact to Application	Cost of Local Recovery	Action
Application segment (user table, index, cluster)	Localized	Low	Low	Use RMAN Block Media Recovery or Re-Create Objects Manually
TEMPORARY segment or temporary table	N/A	N/A	N/A	No impact to permanent objects. Re-create temporary tablespace if required.

The proper recovery method to use depends on the following criteria, as indicated in Table 10–3 and Table 10–4:

- Object affected: The recovery actions available depend on which objects are affected by the corruption. Possible values are:
  - Data dictionary or UNDO segment: The object affected is owned by SYS or is part of UNDO tablespace
  - Temporary segment: Corruptions within a temporary segment or object within a temporary tablespace do not affect permanent objects
  - Application segment: Table, index, or cluster used by the application
- Impact to application

  An object may be critical for the application to function. This includes objects that are critical for the performance and usability of the application. It could also be a history, reporting, or logging table, which may not be as critical. It could also be an object that is no longer in use or a temporary segment.

  This criterion only applies to a Data Guard environment and should be used to decide between recovering the affected object locally and using Data Guard failover. Possible values are:
  - High: Object is critical to the application such that available or performance suffers significantly
  - Low: Object is not critical and has little or no impact to the application
- Cost of local recovery

This criterion only applies to a Data Guard environment and should be used to decide between recovering the affected object locally and using Data Guard failover. This is not a business cost which is assumed to be implicit in deciding how critical an object is to the application but cost in terms of feasibility of recovery, resources required and their impact on performance and total time taken.

Cost of local recovery should include the time to restore and recover the object from a valid source; the time to recover other dependent objects like indexes, constraints, and related tables and its indexes and constraints; availability of resources like disk space, data or index tablespace, temporary tablespace; and impact on performance and functionality of current normal application functions due to absence of the corrupt object.

- Extent of corruption

  Corruption may be localized so that it affects a known number of blocks within one or a few objects, or it may be widespread so that it affects a large portion of an object.

Recovering From Media Failure

When media failure occurs, follow these steps:

1. Determine the Extent of the Media Failure
2. Replace or Move Away From Faulty Hardware
3. Decide Which Recovery Action to Take

Determine the Extent of the Media Failure Use the following methods to determine the extent of the media failure:

- Gather details from error messages

  Gather the file number and file name from the error messages reported. Typical error messages are ORA-1157, ORA-1110 and ORA-1115, ORA-1110.

  For example, from the following error message:

  ```
  ORA-01115: IO error reading block from file 22 (block # 12698)
  ORA-01110: data file 22: '/oradata/SALES/users01.dbf'
  ```

- Check log files for additional error messages

Record additional error messages that appear in the system logs, volume manager logs, alert log, Oracle trace files, or application logs. Note that log files may be distributed across the data server, middle tier, and the client machines.

Replace or Move Away From Faulty Hardware If there is a hardware fault or a suspect component, then it is sensible to either repair the problem or make disk space available on a separate disk subsystem before proceeding with a recovery option.

If there are multiple errors, if there are operating system-level errors against the affected file, or if the errors are transient and keep moving about, then there is little point in proceeding until the underlying problem has been addressed or space is available on alternative disks. Ask your hardware vendor to verify system integrity.

Decide Which Recovery Action to Take The appropriate recovery action depends on what type of file is affected by the media failure. Table 10–5 shows the type of file and the appropriate recovery.

Table 10–5 Recovery Actions for Failure of Different Types of Files

Type of File	Recovery Action
Datafile	Media failure of a datafile is resolved in the same manner in which widespread datafile block corruption is handled. See Also: "Recovering From Datafile Block Corruption" on page 10-24
Control file	Loss of a control file causes the primary database to shut down. The steps to recover from control file failure include making a copy of a good control file, restoring a backup copy of the control file, or manually creating the control file with the `CREATE CONTROLFILE` statement. The proper recovery method depends on the following: ▪ Whether all current control files were lost or just a member of a multiplexed control file ▪ Whether or not a backup control file is available **See Also**: "Performing User-Managed Flashback and Recovery" in *Oracle Database Backup and Recovery Advanced User's Guide*
Standby control file	Loss of a standby control file causes the standby database to shut down. It may also, depending on the primary database protection mode, cause the primary database to shut down. To recover from a standby control file failure, a new standby control file must be created from the primary database and transferred to the standby system. **See Also**: "Creating a Physical Standby Database" in *Oracle Data Guard Concepts and Administration*

Table 10–5 Recovery Actions for Failure of Different Types of Files (Cont.)

Type of File	Recovery Action
Online redo log file	If a media failure has affected the online redo logs of a database, then the appropriate recovery procedure depends on the following: ■ The configuration of the online redo log: mirrored or non-mirrored ■ The type of media failure: temporary or permanent ■ The status of the online redo log files affected by the media failure: current, active, unarchived, or inactive **See Also**: "Advanced User-Managed Recovery Scenarios" in *Oracle Database Backup and Recovery Advanced User's Guide* If the online redo log failure causes the primary database to shut down and incomplete recovery must be used to make the database operational again, then Flashback Database can be used instead of restoring all datafiles. Use Flashback Database to take the database back to an SCN before the SCN of the lost online redo log group. The resetlogs operation that is done as part of the Flashback Database procedure reinitializes all online redo log files. Using Flashback Database is faster than restoring all datafiles. If the online redo log failure causes the primary database to shut down in a Data Guard environment, it may be desirable to perform a Data Guard failover to reduce the time it takes to restore service to users and to reduce the amount of data loss incurred (when using the proper database protection mode). The decision to perform a failover (instead of recovering locally at the primary site with Flashback Database, for example) depends on the estimated time to recover from the outage at the primary site, the expected amount of data loss, and the impact the recovery procedures taken at the primary site may have on the standby database. For example, if the decision is to recover at the primary site, then the recovery steps may require a Flashback Database and open resetlogs, which may incur a full redo log file of lost data. A standby database will have less data loss in most cases than recovering at the primary site because all redo data is available to the standby database. If recovery is done at the primary site and the standby database is ahead of the point to which the primary database is recovered, then the standby database must be re-created or flashed back to a point before the resetlogs SCN on the primary database. **See Also**: "Creating a Physical Standby Database" in *Oracle Data Guard Concepts and Administration*
Standby redo log file	Standby redo log failure affects only the standby database in a Data Guard environment. Most standby redo log failures are handled automatically by the standby database without affecting the primary database. However, if a standby redo log file fails while being archived to, then the primary database treats it as a log archive destination failure. **See Also**: "Determine the Data Protection Mode" on page 7-22

Table 10–5 Recovery Actions for Failure of Different Types of Files (Cont.)

Type of File	Recovery Action
Archived redo log file	Loss of an archived redo log does not affect availability of the primary database directly, but it may significantly affect availability if another media failure occurs before the next scheduled backup, or if Data Guard is being used and the archived redo log had not been fully received by the standby system and applied to the standby database before losing the file. **See Also**: "Advanced User-Managed Recovery Scenarios" in *Oracle Database Backup and Recovery Advanced User's Guide* If an archived redo log is lost in a Data Guard environment and the log has already been applied to the standby database, then there is no impact. If there is no valid backup copy of the lost file, then a backup should be taken immediately of either the primary or standby database because the lost log will be unavailable for media recovery that may be required for some other outage. If the lost archived redo log has not yet been applied to the standby database, then a backup copy of the file must be restored and made available to the standby database. If there is no valid backup copy of the lost archived redo log, then the standby database must be reinstantiated from a backup of the primary database taken after the `NEXT_CHANGE#` of the lost log (see `V$ARCHIVED_LOG`).
Server parameter file (`SPFILE`)	Loss of the server parameter file does not affect availability of the database. `SPFILE` is necessary for database startup. With the flash recovery and RMAN `CONTROLFILE AUTOBACKUP` features enabled, restoring a server parameter file from backup is a fast operation. **See Also**: "Performing Recovery" of *Oracle Database Backup and Recovery Basics*
Oracle Cluster Registry (OCR)	Loss of the Oracle Cluster Registry file affects the availability of RAC and Cluster Ready Services. The OCR file can be restored from a physical backup that is automatically created or from an export file that is manually created by using the `ocrconfig` tool. **See Also**: "Administering Storage in Real Application Clusters" in *Oracle Real Application Clusters Administrator's Guide*

Recovery Methods for Data Failures

The following recovery methods can be used in all environments:

- Use RMAN Datafile Media Recovery
- Use RMAN Block Media Recovery
- Re-Create Objects Manually

Always use local recovery methods when Data Guard is not being used. Local recovery methods may also be appropriate in a Data Guard environment. This section also includes the following topic:

- Use Data Guard to Recover From Data Failure

Use RMAN Datafile Media Recovery

Datafile media recovery recovers an entire datafile or set of datafiles for a database by using the RMAN RECOVER command. When a large or unknown number of data blocks are marked media-corrupt and require media recovery, or when an entire file is lost, the affected datafiles must be restored and recovered.

Use RMAN file media recovery when the following conditions are true:

- The number of blocks requiring recovery is large or unknown
- Block media recovery is not available (for example, if incomplete recovery is required, or if only incremental backups are available for the datafile requiring recovery)

> **See Also:** "Advanced User-Managed Recovery Scenarios" in *Oracle Database Backup and Recovery Advanced User's Guide*

Use RMAN Block Media Recovery

Block media recovery (BMR) recovers one or a set of data blocks marked "media corrupt" within a datafile by using the RMAN BLOCKRECOVER command. When a small number of data blocks are marked media corrupt and require media recovery, you can selectively restore and recover damaged blocks rather than whole datafiles. This results in lower mean time to recovery (MTTR) because only blocks that need recovery are restored and only necessary corrupt blocks undergo recovery. Block media recovery minimizes redo application time and avoids I/O overhead during recovery. It also enables affected datafiles to remain online during recovery of the corrupt blocks. The corrupt blocks, however, remain unavailable until they are completely recovered.

Use block media recovery when:

- A small number of blocks require media recovery and the blocks that need recovery are known. If a significant portion of the datafile is corrupt, or if the amount of corruption is unknown, then a different recovery method should be used.
- Blocks are marked corrupt (verified with the RMAN BACKUP VALIDATE command) and only when complete recovery is required.

Block media recovery cannot be used to recover from the following:

- User error or software bugs that cause logical corruption where the data blocks are intact. See "Recovering from User Error with Flashback Technology" on page 10-35 for additional details for this type of recovery.
- Changes caused by corrupt redo data. Block media recovery requires that all available redo data be applied to the blocks being recovered.

The following are useful practices when using block media recovery:

- The flash recovery area should have a retention policy such that backups of all datafiles are retained on disk longer than the frequency of use of an Oracle tool that detects block corruption. For example, if RMAN `BACKUP VALIDATE` is run once a week, then the flash recovery area should have a retention policy greater than one week. This ensures that corrupt blocks can be recovered quickly using block media recovery with disk-based backups.
- Even if archived redo logs have missing redo data, block media recovery can still work if the block has been renewed since the data file backup used for the restoration of the block, or if there are no changes for the specific block in the missing redo data.
- If RMAN block validation has been run proactively, then the `V$DATABASE_BLOCK_CORRUPTION` view has a list of blocks validated as corrupt by RMAN. RMAN can be instructed to recover all corrupt blocks listed in `V$DATABASE_BLOCK_CORRUPTION` using block media recovery.

> **See Also:** *Oracle Database Backup and Recovery Advanced User's Guide*

Re-Create Objects Manually

Some database objects, such as small look-up tables or indexes, can be recovered quickly by manually re-creating the object instead of doing media recovery.

Use manual object re-creation when:

- You need to re-create a small index because of media corruption. Creating an index online enables the base object to be used concurrently.
- You need to re-create a look-up table or when the scripts to re-create the table are readily available. Dropping and re-creating the table may be the fastest option.

Use Data Guard to Recover From Data Failure

Failover is the operation of taking the production database offline on one site and bringing one of the standby databases online as the new production database. A database switchover is a planned transition in which a standby database and a production database switch roles.

Use Data Guard switchover or failover for data failure when:

- The database is down or when the database is up but the application is unavailable because of data failure, and the time to restore and recover locally is long or unknown.
- The business cost of a switchover or failover and re-creating the standby database is less than the cost of the expected downtime or reduced capacity while local recovery steps are taken.
- The proper decision to recover locally is unclear or too complex, and the data failure is affecting the availability of the application.

See Also: "Database Failover" on page 10-10 and "Database Switchover" on page 10-12

Recovering from User Error with Flashback Technology

Oracle flashback technology revolutionizes data recovery. In the past it took seconds to damage a database but hours to days to recover it. With flashback technology, the time to correct errors can be as short as the time it took to make the error. Fixing user errors that require rewinding the database, table, transaction, or row level changes to a previous point in time is easy and does not require any database or object restoration. Flashback technology provides fine-grained analysis and repair for localized damage such as erroneous row deletion. Flashback technology also enables correction of more widespread damage such as accidentally running the wrong application batch job. Furthermore, flashback technology is exponentially faster than a database restoration.

Flashback technologies are applicable only to repairing the following user errors:

- Erroneous or malicious update, delete or insert transactions
- Erroneous or malicious `DROP TABLE` statements
- Erroneous or malicious batch job or wide-spread application errors

Flashback technologies cannot be used for media or data corruption such as block corruption, bad disks, or file deletions. See "Recovery Solutions for Data Failures" on page 10-22 and "Database Failover" on page 10-10 to repair these outages.

Table 10–6 summarizes the flashback solutions for each type of outage.

Table 10–6 Flashback Solutions for Different Outages

Impact of Outage	Examples of User Errors	Flashback Solutions
Row or transaction See Also: "Resolving Row and Transaction Inconsistencies" on page 10-37	▪ Accidental deletion of row ▪ Erroneous transaction	Use a combination of: ▪ Flashback Query ▪ Flashback Version Query ▪ Flashback Transaction Query **See Also:** "Flashback Query" on page 10-38
Table See Also:"Resolving Table Inconsistencies" on page 10-41	▪ Dropped table ▪ Erroneous transactions affecting one table or a set of tables	▪ Flashback Drop ▪ Flashback Table
Tablespace or database See Also: "Resolving Database-Wide Inconsistencies" on page 10-42	▪ Erroneous batch job affecting many tables or an unknown set of tables ▪ Series of database-wide malicious transactions ▪ Drop tablespace without removing the physical datafiles	▪ Flashback Database ▪ Flashback Database and restoring only the dropped datafiles

Table 10–7 summarizes each flashback feature.

Table 10–7 Summary of Flashback Features

Flashback Feature	Description
Flashback Query	Flashback Query enables you to view data at a point in time in the past. It can be used to view and reconstruct lost data that was deleted or changed by accident. Developers can use this feature to build self-service error correction into their applications, empowering end users to undo and correct their errors. **Note:** Changes are propagated to physical and logical standby databases.
Flashback Version Query	Flashback Version Query uses undo data stored in the database to view the changes to one or more rows along with all the metadata of the changes. Note: Changes are propagated to physical and logical standby databases.

Table 10–7 Summary of Flashback Features (Cont.)

Flashback Feature	Description
Flashback Transaction Query	Flashback Transaction Query enables you to examine changes to the database at the transaction level. As a result, you can diagnose problems, perform analysis, and audit transactions. **Note:** Changes are propagated to physical and logical standby databases.
Flashback Drop	Flashback Drop provides a way to restore accidentally dropped tables. **Note:** Changes are propagated to physical standby databases.
Flashback Table	Flashback Table enables you to quickly recover a table to a point in time in the past without restoring a backup. **Note:** Changes are propagated to physical and logical standby databases.
Flashback Database	Flashback Database enables you to quickly return the database to an earlier point in time by undoing all of the changes that have taken place since that time. This operation is fast because you do not need to restore the backups.

Flashback Database uses the Oracle Database flashback logs, while all other features of flashback technology use the Oracle Database unique undo and multiversion read consistency capabilities. See "Configuration Best Practices for the Database" on page 10-2 for configuring flashback technologies to ensure that the resources from these solutions are available at a time of failure.

The rest of this section includes the following topics:

- Resolving Row and Transaction Inconsistencies
- Resolving Table Inconsistencies
- Resolving Database-Wide Inconsistencies

> **See Also:** *Oracle Database Administrator's Guide*, *Oracle Database Backup and Recovery Basics*, and *Oracle Database Concepts* for more information about flashback technology and automatic undo management

Resolving Row and Transaction Inconsistencies

Resolving row and transaction inconsistencies may require a combination of Flashback Query, Flashback Version Query, Flashback Transaction Query, and the suggested undo statements to rectify the problem. The following sections describe a general approach using a human resources example to resolve row and transaction inconsistencies caused by erroneous or malicious user errors.

This section includes the following topics:

- Flashback Query
- Flashback Version Query
- Flashback Transaction Query
- Example: Using Flashback Technology to Investigate Salary Discrepancy

Flashback Query

Flashback Query, a feature introduced in the Oracle9*i* Database, enables an administrator or user to query any data at some point in time in the past. This powerful feature can be used to view and reconstruct data that may have been deleted or changed by accident. For example:

```
SELECT * FROM EMPLOYEES
       AS OF TIMESTAMP
       TO_DATE('28-Aug-03 14:00','DD-Mon-YY HH24:MI')
 WHERE ...
```

This partial statement displays rows from the EMPLOYEES table starting from 2 p.m. on August 28, 2003. Developers can use this feature to build self-service error correction into their applications, empowering end users to undo and correct their errors without delay, rather than burdening administrators to perform this task. Flashback Query is very simple to manage, because the database automatically keeps the necessary information to reconstruct data for a configurable time into the past.

Flashback Version Query

Flashback Version Query provides a way to view changes made to the database at the row level. It is an extension to SQL and enables the retrieval of all the different versions of a row across a specified time interval. For example:

```
SELECT * FROM EMPLOYEES
       VERSIONS BETWEEN TIMESTAMP
       TO_DATE('28-Aug-03 14:00','dd-Mon-YY hh24:mi') AND
       TO_DATE('28-Aug-03 15:00','dd-Mon-YY hh24:mi')
WHERE ...
```

This statement displays each version of the row, each entry changed by a different transaction, between 2 and 3 p.m. today. A DBA can use this to pinpoint when and how data is changed and trace it back to the user, application, or transaction. This

enables the DBA to track down the source of a logical corruption in the database and correct it. It also enables application developers to debug their code.

Flashback Transaction Query

Flashback Transaction Query provides a way to view changes made to the database at the transaction level. It is an extension to SQL that enables you to see all changes made by a transaction. For example:

```
SELECT UNDO_SQL
FOMR DBA_TRANSACTION_QUERY
WHERE XID = '000200030000002D';
```

This statement shows all of the changes that resulted from this transaction. In addition, compensating SQL statements are returned and can be used to undo changes made to all rows by this transaction. Using a precision tool like this, the DBA and application developer can precisely diagnose and correct logical problems in the database or application.

Example: Using Flashback Technology to Investigate Salary Discrepancy

Consider a human resources (HR) example involving the SCOTT schema. the HR manager reports to the DBA that there is a potential discrepancy in Ward's salary. Sometime before 9:00 a.m., Ward's salary was increased to $1875. The HR manager is uncertain how this occurred and wishes to know when the employee's salary was increased. In addition, he has instructed his staff to reset the salary to the previous level of $1250, and this was completed around 9:15 a.m.

The following steps show how to approach the problem.

1. Assess the problem.

   Fortunately, the HR manager has provided information about the time when the change occurred. We can query the information as it was at 9:00 a.m. with Flashback Query.

   ```
   SELECT EMPNO, ENAME, SAL
   FROM EMP
   AS OF TIMESTAMP TO_DATE('03-SEP-03 09:00','dd-Mon-yy hh24:mi')
   WHERE ENAME = 'WARD';

        EMPNO ENAME             SAL
   ---------- ---------- ----------
         7521 WARD             1875
   ```

We can confirm we have the correct employee by the fact that Ward's salary was $1875 at 09:00 a.m. Rather than using Ward's name, we can now use the employee number for subsequent investigation.

2. Query past rows or versions of the data to acquire transaction information.

   Although it is possible to restrict the row version information to a specific date or SCN range, we decide to query all the row information that we have available for the employee WARD using Flashback Version Query.

```
SELECT EMPNO, ENAME, SAL, VERSIONS_STARTTIME, VERSIONS_ENDTIME
FROM EMP
VERSIONS BETWEEN TIMESTAMP MINVALUE AND MAXVALUE
WHERE EMPNO = 7521
ORDER BY NVL(VERSIONS_STARTSCN,1);

    EMPNO ENAME             SAL VERSIONS_STARTTIME     VERSIONS_ENDTIME
-------- ---------- ---------- ---------------------- ----------------------
     7521 WARD             1250 03-SEP-03 08.48.43 AM  03-SEP-03 08.54.49 AM
     7521 WARD             1875 03-SEP-03 08.54.49 AM  03-SEP-03 09.10.09 AM
     7521 WARD             1250 03-SEP-03 09.10.09 AM
```

   We can see that WARD's salary was increased from $1250 to $1875 at 08:54:49 the same morning and was subsequently reset to $1250 at approximately 09:10:09.

   Also, we can modify the query to determine the transaction information for each of the changes effecting WARD using a similar Flashback Version Query. This time we use the VERSIONS_XID pseudocolumn.

```
SELECT EMPNO, ENAME, SAL, VERSIONS_XID
FROM EMP
VERSIONS BETWEEN TIMESTAMP MINVALUE AND MAXVALUE
WHERE EMPNO = 7521
ORDER BY NVL(VERSIONS_STARTSCN,1);

     EMPNO ENAME             SAL VERSIONS_XID
---------- ---------- ---------- ----------------
      7521 WARD             1250 0006000800000086
      7521 WARD             1875 0009000500000089
      7521 WARD             1250 000800050000008B
```

3. Query the erroneous transaction and the scope of its impact.

With the transaction information (VERSIONS_XID pseudocolumn), we can now query the database to determine the scope of the transaction, using Flashback Transaction Query.

```
SELECT UNDO_SQL
FROM FLASHBACK_TRANSACTION_QUERY
WHERE XID = HEXTORAW('0009000500000089');

UNDO_SQL
--------------------------------------------------------------------------------
update "SCOTT"."EMP" set "SAL" = '950' where ROWID = 'AAACV4AAFAAAAKtAAL';
update "SCOTT"."EMP" set "SAL" = '1500' where ROWID = 'AAACV4AAFAAAAKtAAJ';
update "SCOTT"."EMP" set "SAL" = '2850' where ROWID = 'AAACV4AAFAAAAKtAAF';
update "SCOTT"."EMP" set "SAL" = '1250' where ROWID = 'AAACV4AAFAAAAKtAAE';
update "SCOTT"."EMP" set "SAL" = '1600' where ROWID = 'AAACV4AAFAAAAKtAAB';

6 rows selected.
```

We can see that WARD's salary was not the only change that occurred in the transaction. The information that was changed for the other four employees at the same time as WARD can now be passed back to the HR manager for review.

4. Determine if the corrective statements should be executed.

   If the HR manager decides that the corrective changes suggested by the UNDO_SQL column are correct, then the DBA can execute these statements individually.

Resolving Table Inconsistencies

Oracle provides a FLASHBACK DROP statement to recover from an accidental DROP TABLE statement, and a FLASHBACK TABLE statement to restore a table to a previous point in the database.

This section includes the following topics:

- Flashback Table
- Flashback Drop

Flashback Table

Flashback Table provides the DBA the ability to recover a table, or a set of tables, to a specified point in time quickly and easily. In many cases, Flashback Table alleviates the need to perform more complicated point in time recovery operations. For example:

```
FLASHBACK TABLE orders, order_items
      TO TIMESTAMP
      TO_DATE('29-AUG-03 14.00.00','dd-Mon-yy hh24:mi:ss');
```

This statement rewinds any updates to the ORDERS and ORDER_ITEMS tables that have been done between the current time and specified timestamp in the past. Flashback Table performs this operation online and in place, and it maintains referential integrity constraints between the tables.

Flashback Drop

Dropping or deleting database objects by accident is a common mistake. Users soon realize their mistake, but by then it is too late and there has been no way to easily recover the dropped tables and its indexes, constraints, and triggers. Objects once dropped were dropped forever. Loss of very important tables or other objects (like indexes, partitions or clusters) required DBAs to perform a point-in-time recovery, which can be time-consuming and lead to loss of recent transactions.

Flashback Drop provides a safety net when dropping objects in Oracle Database 10*g*. When a user drops a table, Oracle places it in a recycle bin. Objects in the recycle bin remain there until the user decides to permanently remove them or until space limitations begin to occur on the tablespace containing the table. The recycle bin is a virtual container where all dropped objects reside. Users can look in the recycle bin and undrop the dropped table and its dependent objects. For example, the employees table and all its dependent objects would be undropped by the following statement:

```
FLASHBACK TABLE employees TO BEFORE DROP;
```

Resolving Database-Wide Inconsistencies

Oracle provides Flashback Database to rewind the entire database to a previous point in time. This section includes the following topics:

- Flashback Database
- Using Flashback Database to Repair a Dropped Tablespace

Flashback Database

To bring an Oracle database to a previous point in time, the traditional method is point-in-time recovery. However, point-in-time recovery can take hours or even days, since it requires the whole database to be restored from backup and recovered to the point in time just before the error was introduced into the database. With the

size of databases constantly growing, it will take hours or even days just to restore the whole database.

Flashback Database is a new strategy for doing point-in-time recovery. It quickly rewinds an Oracle database to a previous time to correct any problems caused by logical data corruption or user error. Flashback logs are used to capture old versions of changed blocks. One way to think of it is as a continuous backup or storage snapshot. When recovery needs to be performed the flashback logs are quickly replayed to restore the database to a point in time before the error and just the changed blocks are restored. It is extremely fast and reduces recovery time from hours to minutes. In addition, it is easy to use. A database can be recovered to 2:05 p.m. by issuing a single statement. Before the database can be recovered, all instances of the database must be shut down and one of the instances subsequently mounted. The following is an example of a FLASHBACK DATABASE statement.

```
FLASHBACK DATABASE TO TIMESTAMP TIMESTAMP'2002-11-05 14:00:00';
```

No restoration from tape, no lengthy downtime, and no complicated recovery procedures are required to use it. You can also use Flashback Database and then open the database in read-only mode and examine its contents. If you determine that you flashed back too far or not far enough, then you can reissue the FLASHBACK DATABASE statement or continue recovery to a later time to find the proper point in time before the database was damaged. Flashback Database works with a production database, a physical standby database, and a logical standby database.

These steps are recommended for using Flashback Database:

1. Determine the time or the SCN to which to flash back the database.
2. Verify that there is sufficient flashback log information.

   ```
   SELECT OLDEST_FLASHBACK_SCN,
     TO_CHAR(OLDEST_FLASHBACK_TIME, 'mon-dd-yyyy HH:MI:SS')
     FROM V$FLASHBACK_DATABASE_LOG;
   ```

3. Flash back the database to a specific time or SCN. (The database must be mounted to perform a Flashback Database.)

   ```
   FLASHBACK DATABASE TO SCN scn;
   ```

   or

   ```
   FLASHBACK DATABASE TO TIMESTAMP TO_DATE date;
   ```

4. Open the database in read-only mode to verify that it is in the correct state.

```
ALTER DATABASE OPEN READ ONLY;
```

If more flashback data is required, then issue another FLASHBACK DATABASE statement. (The database must be mounted to perform a Flashback Database.)

If you want to move forward in time, issue a statement similar to the following:

```
RECOVER DATABASE UNTIL [TIME | CHANGE] date | scn;
```

5. Open the database.

Caution:

After you open the database, you will not be able to flash back to an SCN before the resetlogs SCN, so ensure that the database is in the correct state before issuing the ALTER DATABASE OPEN RESETLOGS statement.

```
ALTER DATABASE OPEN RESETLOGS;
```

Other considerations when using Flashback Database are as follows:

- If there are not sufficient flashback logs, then use Data Guard if available or restore from backups.
- After flashing back a database, any dependent database such as a standby database needs to be flashed back. See Chapter 11, "Restoring Fault Tolerance".

Using Flashback Database to Repair a Dropped Tablespace

Flashback Database does not automatically fix this problem, but it can be used to dramatically reduce the downtime. You can flash back the production database to a point before the tablespace was dropped and then restore a backup of the corresponding datafiles from the affected tablespace and recover to a time before the tablespace was dropped.

Follow these recommended steps to use Flashback Database to repair a dropped tablespace:

1. Determine the SCN or time you dropped the tablespace.
2. Flash back the database to a time before the tablespace was dropped. You can use a statement similar to the following:

```
FLASHBACK DATABASE TO BEFORE SCN drop_scn;
```

3. Restore, rename, and bring datafiles online.
   a. Restore only the datafiles from the affected tablespace from a backup.
   b. Rename the unnamed files to the backup files.

      ```
      ALTER DATABASE RENAME FILE '.../UNNAMED00005' to 'restored_file';
      ```

   c. Bring the datafiles online.

      ```
      ALTER DATABASE DATAFILE 'name' ONLINE;
      ```

4. Query and recover the database.

   ```
   SELECT CHECKPOINT_CHANGE# FROM V$DATAFILE_HEADER WHERE FILE#=1;
   RECOVER DATABASE UNTIL CHANGE checkpoint_change#;
   ```

5. Open the database.

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

RAC Rolling Upgrade

"One-off" patches or interim patches to database software are usually applied to implement known fixes for software problems an installation has encountered or to apply diagnostic patches to gather information regarding a problem. Such patch application is often carried out during a scheduled maintenance outage.

Oracle now provides the capability to do rolling patch upgrades with Real Application Clusters with little or no database downtime. The tool used to achieve this is the `opatch` command-line utility.

The advantage of a RAC rolling upgrade is that it enables at least some instances of the RAC installation to be available during the scheduled outage required for patch upgrades. Only the RAC instance that is currently being patched needs to be brought down. The other instances can continue to remain available. This means that the impact on the application downtime required for such scheduled outages is further minimized. Oracle's `opatch` utility enables the user to apply the patch successively to the different instances of the RAC installation.

Rolling upgrade is available only for patches that have been certified by Oracle to be eligible for rolling upgrades. Typically, patches that can be installed in a rolling upgrade include:

- Patches that do not affect the contents of the database such as the data dictionary

- Patches not related to RAC internode communication
- Patches related to client-side tools such as SQL*PLUS, Oracle utilities, development libraries, and Oracle Net
- Patches that do not change shared database resources such as datafile headers, control files, and common header definitions of kernel modules

Rolling upgrade of patches is currently available for one-off patches only. It is not available for patch sets.

Rolling patch upgrades are not available for deployments where the Oracle Database software is shared across the different nodes. This is the case where the Oracle home is on Cluster File System (CFS) or on shared volumes provided by file servers or NFS-mounted drives. The feature is only available where each node has its own copy of the Oracle Database software.

This section includes the following topics:

- Applying a Patch with opatch
- Rolling Back a Patch with opatch
- Using opatch to List Installed Software Components and Patches
- Recommended Practices for RAC Rolling Upgrades

Applying a Patch with opatch

The `opatch` utility applies a patch successively to nodes of the RAC cluster. The nature of the patch enables a RAC installation to run in a mixed environment. Different instances of the database may be operating at the same time, and the patch may have been applied to some instances and not others. The `opatch` utility automatically detects the nodes of the cluster on which a specific RAC deployment has been implemented. The patch is applied to each node, one at a time. For each node, the DBA is prompted to shut down the instance. The patch is applied to the database software install on that node. After the current node has been patched, the instance can be restarted. After the patch is applied on the current node, the DBA is allowed to choose the next RAC node to apply the patch to. The cycle of instance shutdown, patch application, and instance startup is repeated. Thus, at any time during the patch application, only one node needs to be down.

To check if a patch is a rolling patch, execute the following on UNIX platforms:

```
opatch query -is_rolling
```

(On Windows, execute `opatch.bat`.)

Enter the patch location after the prompt.

To apply a patch to all nodes of the RAC cluster, execute the following command:

```
opatch apply patch_location
```

opatch automatically recognizes the patch to be a rolling patch and provides the required behavior.

To apply a patch to only the local node, enter the following command:

```
opatch apply -local patch_location
```

To check the results of a patch application, check the logs in the following location:

```
$ORACLE_HOME/.patch_storage/patch_id/patch_id_Apply_timestamp.log
```

Rolling Back a Patch with opatch

Patches can be rolled back with the opatch utility. This enables the DBA to remove a troublesome patch or a patch that is no longer required. This can be done as a rolling procedure.

To roll back a patch across all nodes of a RAC cluster, execute the following command:

```
opatch rollback -id patch_id -ph patch_location
```

To roll back a patch on the local node only, enter the following command:

```
opatch rollback -local -id patch_id -ph patch_location
```

To check the results of a patch rollback, check the logs in the following location:

```
$ORACLE_HOME/.patch_storage/patch_id/patch_id_RollBack_timestamp.log
```

Using opatch to List Installed Software Components and Patches

The opatch utility also provides an option to list the installed software components as well as the installed patches. Enter the following command:

```
opatch lsinventory
```

For details on usage and the other options to these commands, see MetaLink Notes 242993.1 and 189489.1 at http://metalink.oracle.com.

Recommended Practices for RAC Rolling Upgrades

The following are recommended practices for all database patch upgrades:

- Always confirm with Oracle Support Services that the patch is valid for your problem and for your deployment environment.
- Have a plan for applying the patch as well as a plan for backing out the patch.
- Apply the patch to your test environment first and verify that it fixes the problem.
- When you plan the elapsed time for applying the patch, include time for starting up and shutting down the other tiers of your technology stack if necessary.

The following are additional recommended practices for RAC rolling upgrades.

- If multiple instances share an Oracle home, then all of them will be affected by application of a patch. The DBA should verify that this will not cause unintentional side effects. Also, all such instances on a node must be shut down during the patch application. Scheduled outage planning should take this into account. As a best practice, only similar applications should share an Oracle home on a node. This provides greater flexibility for patching.
- The Oracle inventory on each node is a repository of the Oracle Database software installed on the node. The inventory is node-specific. It is shared by all Oracle software installed on the node. It is similar across nodes only if all nodes are exactly the same in terms of the Oracle Database software deployed, the deployment configuration, and patch levels. Because the Oracle inventory greatly aids the patch application and patch management process, it is recommended that its integrity be maintained. Oracle inventory should be backed up after each patch installation to any Oracle software on a specific node. This applies to the Oracle inventory on each node of the cluster.
- Use the Oracle Universal Installer to install all Oracle database software. This creates the relevant repository entries in the Oracle inventory on each node of the cluster. Also, use the Oracle Universal Installer to add nodes to an existing RAC cluster.

  However, if this was not done or is not feasible for some reason, adding information about an existing Oracle database software installation to the Oracle inventory can be done with the `attach` option of the `opatch` utility. Node information can be also added with this option.
- The nature of the rolling patch upgrade enables it to be applied to only some nodes of the RAC cluster. So an instance can be operating with the patch

applied, while another instance is operating without the patch. This is not possible for nonrolling patch upgrades. Apply nonrolling patch upgrades to all instances before the RAC deployment is activated. A mixed environment is useful if a patch needs to be tested before deploying it to all the instances. Applying the patch with the `-local` option is the recommended way to do this.

In the interest of keeping all instances of the RAC cluster at the same patch level, it is strongly recommended that after a patch has been validated, it should be applied to all nodes of the RAC installation. When instances of a RAC cluster have similar patch software, services can be migrated among instances without running into the problem a patch may have fixed.

- All patches (including those applied by rolling upgrades) should be maintained online and not removed once they have been applied. This is useful if a patch needs to be rolled back or needs to be applied again.

  The patches should be stored in a location that is accessible by all nodes of the cluster. Thus all nodes of the cluster are equivalent in their capability to apply or roll back a patch.

- Rolling patch upgrades, just like any other patch upgrade, should be done when no other patch upgrade or Oracle installation is being done on the node. Application of multiple patches is a sequential process. The scheduled outage should be planned accordingly.

- If multiple patches need to be applied and they must be applied at the same time, and if only some of these patches are eligible for rolling upgrade, apply all of them in a nonrolling manner. This reduce the overall time required to get through the patching process.

- For patches that are not eligible for rolling upgrade, the next best option for RAC deployments is the `minimize_downtime` option of the `apply` command.

- Perform the rolling upgrade when system usage is low. This ensures minimal disruption of service for the end user.

Upgrade with Logical Standby Database

Using a logical standby database enables you to accomplish upgrades for database software and patch sets with almost no downtime.

Note: This capability is not available for upgrading from Oracle9*i* to Oracle Database 10*g*.

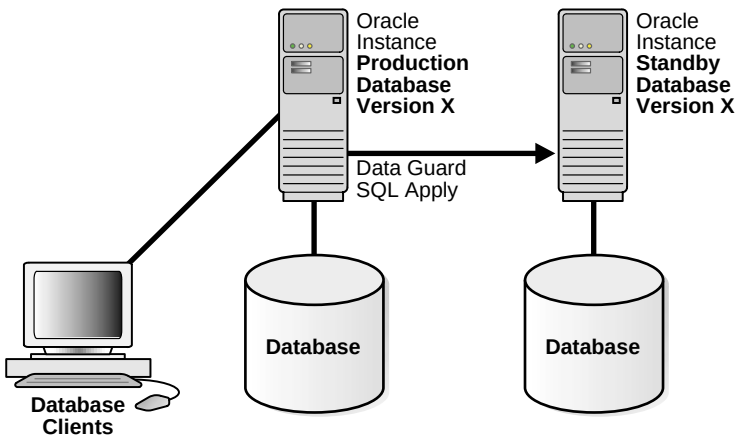
If a logical standby database does not currently exist, then verify that a logical standby database supports all of the essential datatypes of your application.

See Also: *Oracle Data Guard Concepts and Administration* for a list of datatypes supported by the logical standby database

If you cannot use a logical standby database because of the datatypes in your application, then perform the upgrade as documented in *Oracle Database Upgrade Guide*

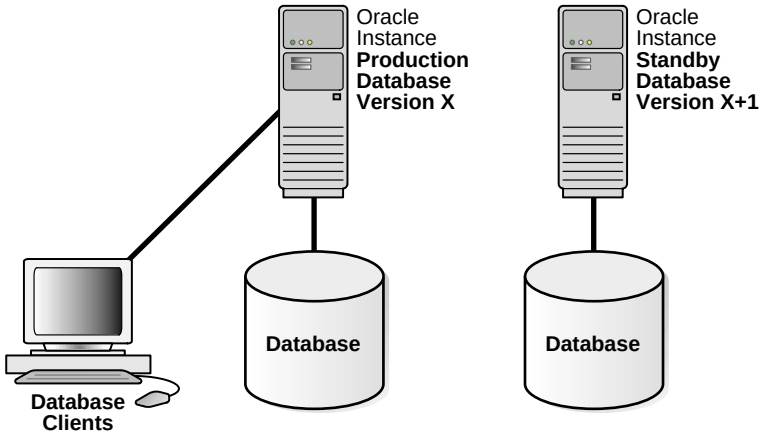
First, create or establish a logical standby database. Figure 10–4 shows a production database and a logical standby database, which are both version X databases.

Figure 10–4 Establish a Logical Standby Database



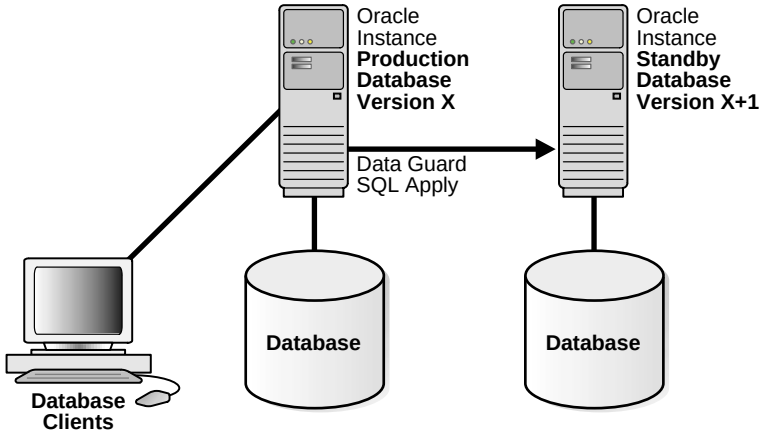
Second, stop the SQL Apply process and upgrade the database software on the logical standby database to version X+1. Figure 10–5 shows the production database, version X, and the logical standby database, version X+1.

Figure 10–5 Upgrade the Logical Standby Database Version



Third, restart SQL Apply and operate with version X on the production database and version X+1 on the standby database. The configuration can run in the mixed mode shown in Figure 10–6 for an arbitrary period to validate the upgrade in the production environment.

Figure 10–6 Running in Mixed Mode

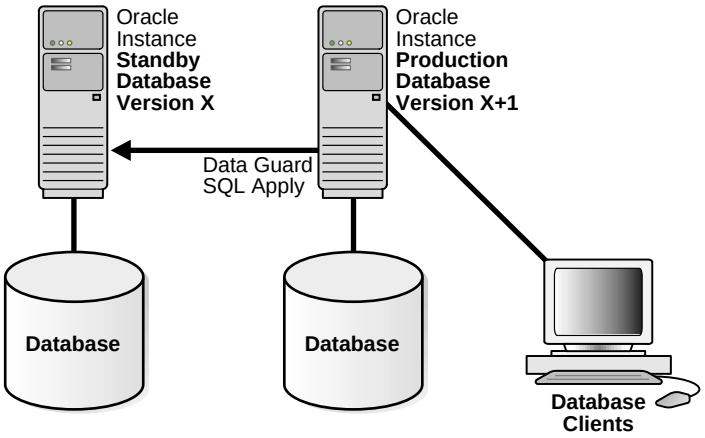


When you are satisfied that the upgraded software is operating properly, you can reverse the database roles by performing a switchover. This may take only a few

seconds. Switch the database clients to the new production database, so that the application becomes active. If application service levels degrade for some reason, then you can open the previous production database again, switch users back, and terminate the previous steps.

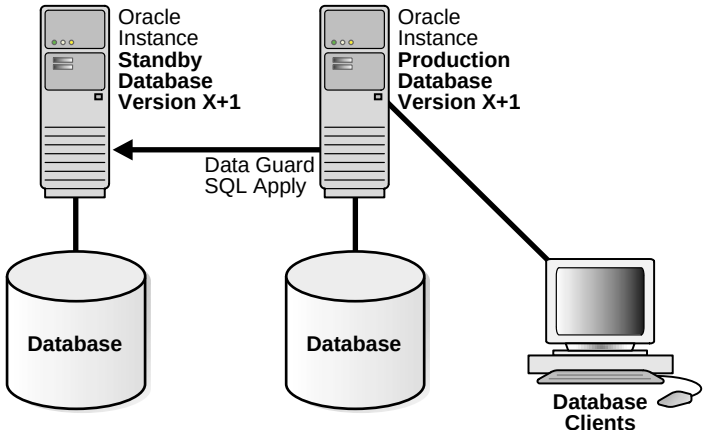
Figure 10–7 shows that the former standby database (version X+1) is now the production database, and the former production database (version X) is now the standby database. The clients are connected to the new production database.

Figure 10–7 After a Switchover Has Been Performed



Upgrade the new standby database. Figure 10–8 shows the system after both databases have been upgraded to version X+1.

Figure 10–8 Both Databases Have Been Upgraded



The role reversal that was just described includes the following steps:

1. Initiate a Data Guard switchover, making the original standby database the production database. This can occur in a few seconds.
2. Switch the database clients to the new production database, so that the application and service become active. If application service levels degrade for some reason, then you can open the previous production database again, switch users back, and terminate the previous steps.
3. Upgrade the database software on the standby database after you are satisfied with current operations.
4. Raise the compatibility level of both databases.

Online Object Reorganization

Oracle's online object reorganization capabilities have been available since Oracle8*i*. These capabilities enable object reorganization to be performed even while the underlying data is being modified.

Table 10–8 describes a few of the object reorganization capabilities available with Oracle Database 10*g*.

Table 10–8 Some Object Reorganization Capabilities

Object Type	Example of Object Reorganization Solution	Description of Solution
Table	`DBMS_REDEFINITION` PL/SQL package	A PL/SQL package that provides a mechanism to redefine tables online
Index	Rebuild index	Rebuild an index that has previously been marked as unusable
Tablespace	Rename tablespace	Enables an existing tablespace to be renamed without rebuilding the tablespace and its contents

Online Table Reorganization

In highly available systems, it is occasionally necessary to redefine large tables that are constantly accessed to improve the performance of queries or DML performed. Oracle provides the `DBMS_REDEFINITION` PL/SQL package to redefine tables online. This package provides a significant increase in availability compared to traditional methods of redefining tables that require tables to be taken offline.

See Also: *Oracle Database Administrator's Guide*

Online Index Reorganization

An index can be rebuilt online using the previous index definition, optionally moving the index to a new tablespace.

See Also: *Oracle Database Administrator's Guide*

Online Tablespace Reorganization

Oracle Database 10*g* introduces the ability to rename a tablespace, similar to the ability to rename a column, table and datafile. Previously, the only way to change a tablespace name was to drop and re-create the tablespace, but this meant that the contents of the tablespace had to be dropped and rebuilt later. With the ability to rename a tablespace online, there is no interruption to the users.

```
ALTER TABLESPACE USERS RENAME TO new_tablespace_name;

Tablespace altered.
```

See Also: *Oracle Database Administrator's Guide*

11

Restoring Fault Tolerance

This chapter describes how to restore redundancy to your environment after a failure. It includes the following topics:

- Restoring Full Tolerance
- Restoring Failed Nodes or Instances in a RAC Cluster
- Restoring the Standby Database After a Failover
- Restoring Fault Tolerance after Secondary Site or Clusterwide Scheduled Outage
- Restoring Fault Tolerance after a Standby Database Data Failure
- Restoring Fault Tolerance After the Production Database Has Opened Resetlogs
- Restoring Fault Tolerance after Dual Failures

Restoring Full Tolerance

Whenever a component within an HA architecture fails, then the full protection, or fault tolerance, of the architecture is compromised and possible single points of failure exist until the component is repaired. Restoring the HA architecture to full fault tolerance to reestablish full RAC, Data Guard, or MAA protection requires repairing the failed component. While full fault tolerance may be sacrificed during a scheduled outage, the method of repair is well understood because it is planned, the risk is controlled, and it ideally occurs at times best suited for continued application availability. However, for unscheduled outages, the risk of exposure to a single point of failure must be clearly understood.

This chapter describes the steps needed to restore database fault tolerance. It includes the following topics:

For RAC environments:

- Restoring Failed Nodes or Instances in a RAC Cluster

For Data Guard and MAA environments:

- Restoring the Standby Database After a Failover
- Restoring Fault Tolerance after Secondary Site or Clusterwide Scheduled Outage
- Restoring Fault Tolerance after a Standby Database Data Failure
- Restoring Fault Tolerance after Dual Failures

Note: If you were using the Data Guard Broker before the failure occurred, then you must remove the configuration through Enterprise Manager or the Data Guard command-line interface before modifying initialization parameters that are related to Data Guard. Re-create the Data Guard Broker configuration after restoring fault tolerance.

Restoring Failed Nodes or Instances in a RAC Cluster

Ensuring that application services fail over quickly and automatically within a RAC cluster, or between primary and secondary sites, is important when planning for both scheduled and unscheduled outages. It is also important to understand the steps and processes for restoring failed instances or nodes within a RAC cluster or

databases between sites, to ensure that the environment is restored to full fault tolerance after any errors or issues are corrected.

Adding a failed node back into the cluster or restarting a failed RAC instance is easily done after the core problem that caused the specific component to originally fail has been corrected. However, the following are additional considerations.

- When to perform these tasks in order to incur minimal or no impact on the current running environment
- Resetting network components (such as load balancer) which were modified for failover and now need to be reset
- Failing back or rebalancing existing connections

How an application runs within a RAC environment (similar to initial failover) also dictates how to restore the node or instance, as well as whether to perform other processes or steps.

After the problem that caused the initial node or instance failure has been corrected, a node or instance can be restarted and added back into the RAC environment at any time. However, there may be some performance impact on the current workload when rejoining the node or instance. Table 11–1 summarizes the performance impact of restarting or rejoining a node or instance.

Table 11–1 Performance Impact of Restarting or Rejoining a Node or Instance

Action	Impact on Runtime Application
Restarting a node or rejoining a node into a cluster	There may be some potential performance impact while the reconfiguration occurs to add this node back into the cluster. There may or may not be an impact on the performance of the running application, but this should be evaluated.
Restarting or rejoining a RAC instance	When you restart a RAC instance, there may be some potential performance impact while lock reconfiguration takes place. Evaluation tests show the impact on current applications to be minimal, but they should be evaluated with an appropriate test workload.

Therefore, it is important to consider the following when restoring a node or RAC instance:

- Test and evaluate the performance impact under a stress workload when rejoining a node into the cluster or restarting an Oracle RAC instance.
- If service levels are acceptable and 2 or more RAC instances are still available in the cluster, then consider rejoining the failed instance during nonpeak work periods.

See Also:

- Your vendor-specified cluster management documentation for detailed steps on how to start and join a node back into a cluster
- *Oracle Real Application Clusters Administrator's Guide* for more information about restarting a RAC instance

The rest of this section includes the following topics:

- Recovering Service Availability
- Considerations for Client Connections After Restoring a RAC Instance

Recovering Service Availability

After a failed node has been brought back into the cluster and its instance has been started, RAC's Cluster Ready Services (CRS) automatically manages the virtual IP address used for the node and the services supported by that instance automatically. A particular service may or may not be started for the restored instance. The decision by CRS to start a service on the restored instance depends on how the service is configured and whether the proper number of instances are currently providing access for the service. A service is not relocated back to a preferred instance if the service is still being provided by an available instance to which it was moved by CRS when the initial failure occurred. CRS restarts services on the restored instance if the number of instances that are providing access to a service across the cluster is less than the number of preferred instances defined for the service. After CRS restarts a service on a restored instance, CRS notifies registered applications of the service change.

For example, suppose the HR service is defined with instances A and B as preferred and instances C and D as available in case of a failure. If instance B fails and CRS starts up the HR service on C automatically, then when instance B is restarted, the HR service remains at instance C. CRS does not automatically relocate a service back to a preferred instance.

Suppose a different scenario in which the HR service is defined with instances A, B, C, and D as preferred and no instances defined as available, spreading the service across all nodes in the cluster. If instance B fails, then the HR service remains available on the remaining three nodes. CRS automatically starts the HR service on instance B when it rejoins the cluster because it is running on fewer instances than configured. CRS notifies the applications that the HR service is again available on instance B.

See Also:

- *Oracle Real Application Clusters Administrator's Guide*
- *Oracle Real Application Clusters Deployment and Performance Guide*

Considerations for Client Connections After Restoring a RAC Instance

After a RAC instance has been restored, additional steps may be required, depending on the current resource utilization and performance of the system, the application configuration, and the network load balancing that has been implemented.

Existing connections (which may have failed over or started as a new session) on the surviving RAC instances, are not automatically redistributed or failed back to an instance that has been restarted. Failing back or redistributing users may or may not be necessary, depending on the current resource utilization and the capability of the surviving instances to adequately handle and provide acceptable response times for the workload. If the surviving RAC instances do not have adequate resources to run a full workload or to provide acceptable response times, then it may be necessary to move (disconnect and reconnect) some existing user connections to the restarted instance.

New connections are started as they are needed, on the least-used node, assuming connection load balancing has been configured. Therefore, the new connections are automatically load-balanced over time.

An application service can be:

- Partitioned with services running on a subset of RAC instances
- Nonpartitioned so that all services run equally across all nodes

This is valuable for modularizing application and database form and function while still maintaining a consolidated data set. For the cases where an application is partitioned or has a combination of partitioning and non-partitioning, the response time and availability aspects for each service should be considered. If redistribution or failback of connections for a particular service is required, then you can rebalance workloads manually with the `DBMS_SERVICE.disconnect_session` PL/SQL procedure. You can use this procedure to disconnect sessions associated with a service while the service is running.

For load-balancing application services across multiple RAC instances, Oracle Net connect-time failover and connection load balancing are recommended. This feature does not require changes or modifications for failover or restoration. It is also

possible to use hardware-based load balancers. However, there may be limitations in distinguishing separate application services (which is understood by Oracle Net Services) and restoring an instance or a node. For example, when a node or instance is restored and available to start receiving new connections, a manual step may be required to include the restored node or instance in the hardware-based load balancer logic, whereas Oracle Net Services does not require manual reconfiguration.

Table 11–2 summarizes the considerations for new and existing connections after an instance has been restored. The considerations differ depending on whether the application services are partitioned, nonpartitioned, or have a combination of each type. The actual redistribution of existing connections may or may not be required depending on the resource utilization and response times.

Table 11–2 Restoration and Connection Failback

Application Services	Failback or Restore Existing Connections	Failback or Restore New Connections
Partitioned	Existing sessions are not automatically relocated back to the restored instance. Use `DBMS_SERVICE.disconnect_session` to manually disconnect sessions and allow them to be reestablished on one of the remaining instances that provides the service.	Automatically routes to the restored instance by using the Oracle Net Services configuration.
Nonpartitioned	No action is necessary unless the load needs to be rebalanced, because restoring the instance means that the load there is low. If the load needs to be rebalanced, then the same problems are encountered as if application services were partitioned.	Automatically routes to the restored instance (because its load should be lowest) by using the Oracle Net Services configuration

Figure 11–1 shows a 2-node partitioned RAC database. Each instance services a different portion of the application (HR and Sales). Client processes connect to the appropriate instance based on the service they require.

Figure 11–1 Partitioned 2-Node RAC Database

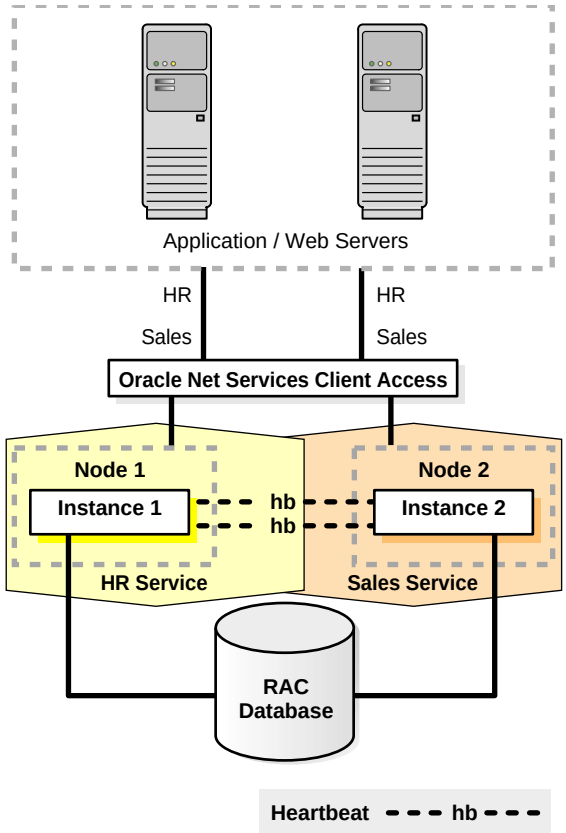
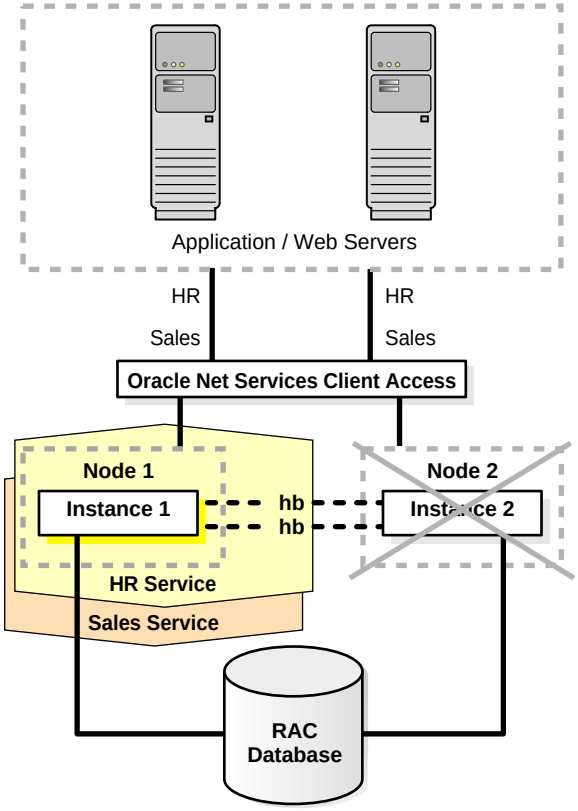


Figure 11–2 shows what happens when one RAC instance fails.

Figure 11–2 RAC Instance Failover in a Partitioned Database



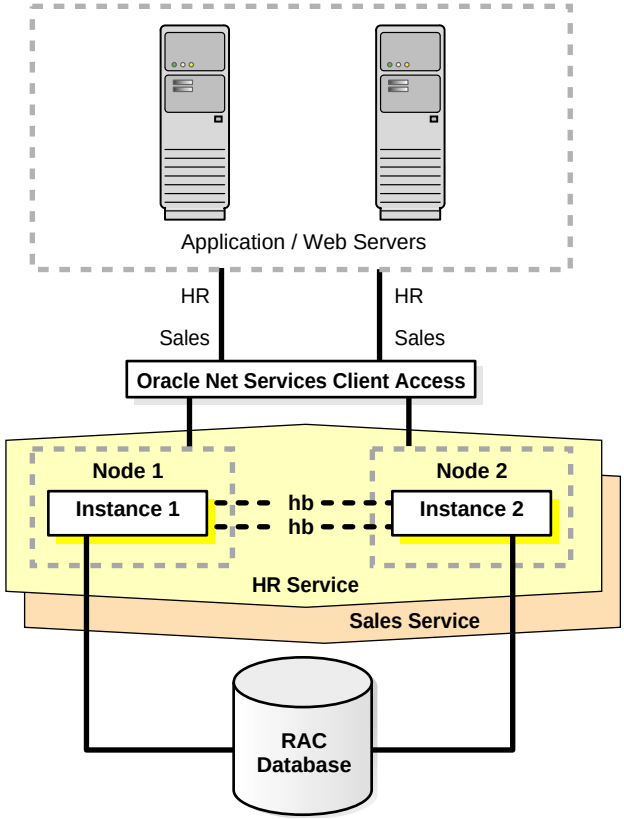
If one RAC instance fails, then the service and existing client connections can be automatically failed over to another RAC instance. In this example, the HR and Sales services are both supported by the remaining RAC instance. In addition, new client connections for the Sales service can be routed to the instance now supporting this service.

After the failed instance has been repaired and restored to the state shown in Figure 11–1 and the Sales service is relocated to the restored instance failed-over clients and any new clients that had connected to the Sales service on the failed-over instance may need to be identified and failed back. New client connections, which are started after the instance has been restored, should automatically connect back to the original instance. Therefore, over time, as older connections disconnect, and new sessions connect to the Sales service, the client

load migrates back to the restored instance. Rebalancing the load immediately after restoration depends on the resource utilization and application response times.

Figure 11–3 shows a nonpartitioned application. Services are evenly distributed across both active instances. Each instance has a mix of client connections for both HR and Sales.

Figure 11–3 Nonpartitioned RAC Instances



If one RAC instance fails, then CRS moves the services that were running on the failed instance. In addition, new client connections are routed only to the available RAC instances that offer that service.

After the failed instance has been repaired and restored to the state shown in Figure 11–3, some clients may need to be moved back to the restored instance. For

nonpartitioned applications, identifying appropriate services is not required for rebalancing the client load among all available instances. Also, this is necessary only if a single instance is not able to adequately service the requests.

New client connections that are started after the instance has been restored should automatically connect back to the restored instance because it has a smaller load. Therefore, over time, as older connections disconnect and new sessions connect to the restored instance, the client load will again evenly balance across all available RAC instances. Rebalancing the load immediately after restoration depends on the resource utilization and application response times.

Restoring the Standby Database After a Failover

Following an unplanned production database outage that requires a failover, full fault tolerance is compromised until the physical or logical standby database is reestablished. Full database protection should be restored as soon as possible. Steps for restoring fault tolerance differ slightly for physical and logical standby databases.

Standby databases do not need to be reinstantiated because of Oracle's Flashback Database feature. Flashback Database:

- Saves hours of database restoration time
- Reduces overall complexity in restoring fault tolerance
- Reduces the time that the system is vulnerable because the standby database is re-created more quickly

This section includes the following topics:

- Restoring a Physical Standby Database After a Failover
- Restoring a Logical Standby Database After a Failover

Restoring a Physical Standby Database After a Failover

The following steps are required to restore a physical standby database after a failover. The steps assume that archived redo logs and sufficient flashback log data are available.

- Step 1P: Retrieve STANDBY_BECAME_PRIMARY_SCN
- Step 2P: Flash Back the Previous Production Database
- Step 3P: Mount New Standby Database From Previous Production Database

- Step 4P: Archive to New Standby Database From New Production Database
- Step 5P: Start Managed Recovery

Step 1P: Retrieve STANDBY_BECAME_PRIMARY_SCN

From the new production database, execute the following query:

```
SELECT TO_CHAR(STANDBY_BECAME_PRIMARY_SCN) FROM V$DATABASE;
```

Use this SCN to convert the previous production database to a standby database.

Step 2P: Flash Back the Previous Production Database

Log on to the previous production database and execute the following statements:

```
SHUTDOWN IMMEDIATE; /*if necessary */
STARTUP MOUNT;
FLASHBACK DATABASE TO SCN standby_became_primary_scn;
```

If there is insufficient flashback data, then see *Oracle Data Guard Concepts and Administration* about creating a new standby database.

Step 3P: Mount New Standby Database From Previous Production Database

Mounting the new standby database requires the following substeps:

1. Turn off flashback mode. This deletes the flashback logs, which are obsolete after the standby control file is restored.

   ```
   ALTER DATABASE FLASHBACK OFF;
   ```

2. Create the standby control file.

   ```
   ALTER DATABASE CREATE STANDBY CONTROLFILE AS controlfile_name;
   SHUTDOWN IMMEDIATE;
   ```

3. Issue operating system copy commands to replace the current control files with the new standby control files.
4. Mount the new standby database with the corresponding standby control file.

   ```
   STARTUP MOUNT;
   ```

5. Ensure that the standby listener is running.

   ```
   LSNRCTL STAT list_name;
   ```

Step 4P: Archive to New Standby Database From New Production Database

Before the new standby database was created, the current production remote standby archive destination probably stopped with an error and is no longer shipping files to the remote destination. To restart remote archiving, you may have to reenable the standby archive destination.

Query the V$ARCHIVE_DEST_STATUS view to see the current state of the archive destinations.

```
SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, ERROR, SRL
    FROM V$ARCHIVE_DEST_STATUS;
```

Enable the remote archive destination.

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
```

Switch a redo log file and verify that it was sent successfully.

```
ALTER SYSTEM SWITCH LOGFILE;
SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, ERROR, SRL
    FROM V$ARCHIVE_DEST_STATUS;
```

Shipping the archived redo log from the new production database notifies the standby database of the new production database incarnation number.

Step 5P: Start Managed Recovery

Start managed recovery or real-time apply managed recovery with one of the following statements:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;
```

or

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
DISCONNECT;
```

Ensure that recovery is applying the archived redo logs.

```
SELECT * FROM V$MANAGED_STANDBY;
```

Step 6P: Restart MRP After It Encounters the End-of-Redo Marker

The managed recovery process (MRP) stops after it encounters the end-of-redo marker that demarcates when the Data Guard failover was completed in the redo stream. This is not an error. Restart MRP, and it will continue with no problem.

Restoring a Logical Standby Database After a Failover

The following steps are required to restore a logical standby database after a failover:

- Step 1L: Retrieve END_PRIMARY_SCN
- Step 2L: Flash Back the Previous Production Database
- Step 3L: Open New Logical Standby Database and Start SQL Apply

Step 1L: Retrieve END_PRIMARY_SCN

On the new production database, query for the SCN at which the previous standby database became the new production database.

```
SELECT VALUE AS TO_CHAR(STANDBY_BECAME_PRIMARY_SCN) FROM DBA_LOGSTDBY_PARAMETERS
WHERE NAME = 'END_PRIMARY_SCN';
```

Step 2L: Flash Back the Previous Production Database

You can create a new logical standby database by mounting the previous production database, flashing it back to STANDBY_BECAME_PRIMARY_SCN, and then enabling the previous guard level setting.

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
FLASHBACK DATABASE TO SCN standby_became_primary_scn;
ALTER DATABASE GUARD [ALL | STANDBY | NONE];
```

Step 3L: Open New Logical Standby Database and Start SQL Apply

```
ALTER DATABASE OPEN RESETLOGS;
ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY dblink;
```

You need to create a database link from the new logical standby database to the new production database if it does not already exist. Use the following syntax:

```
CREATE PUBLIC DATABASE LINK dblink
  CONNECT TO system IDENTIFIED BY password
  USING 'service_name_of_new_primary_database';
```

Restoring Fault Tolerance after Secondary Site or Clusterwide Scheduled Outage

The following steps are required to restore full fault tolerance after a scheduled secondary site or clusterwide outage:

- Step 1: Start the Standby Database
- Step 2: Start Recovery
- Step 3: Verify Log Transport Services on Production Database
- Step 4: Verify that Recovery is Progressing on Standby Database
- Step 5: Restore Production Database Protection Mode

Step 1: Start the Standby Database

You may have to restore the standby database from local backups, local tape backups, or from the primary site backups if the data in the secondary site has been damaged. Re-create the standby database from the new production database by following the steps for creating a standby database in *Oracle Data Guard Concepts and Administration*.

After the standby database has been reestablished, start the standby database.

Type of Standby Database	SQL Statement
Physical	`STARTUP MOUNT;`
Logical	`STARTUP;`

Step 2: Start Recovery

Type of Standby Database	SQL Statement
Physical	`RECOVER MANAGED STANDBY DATABASE DISCONNECT;`
Logical	`ALTER DATABASE START LOGICAL STANDBY APPLY;`

Step 3: Verify Log Transport Services on Production Database

You may have to reenable the production database remote archive destination. Query the V$ARCHIVE_DEST_STATUS view first to see the current state of the archive destinations:

```
SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, ERROR, SRL
FROM V$ARCHIVE_DEST_STATUS;
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
ALTER SYSTEM SWITCH LOGFILE;
```

Verify log transport services between the production and standby databases by checking for errors. Query V$ARCHIVE_DEST and V$ARCHIVE_DEST_STATUS views.

```
SELECT STATUS, TARGET, LOG_SEQUENCE, TYPE, PROCESS, REGISTER, ERROR
    FROM V$ARCHIVE_DEST;
SELECT * FROM V$ARCHIVE_DEST_STATUS WHERE STATUS!='INACTIVE';
```

Step 4: Verify that Recovery is Progressing on Standby Database

For a physical standby database, verify that there are no errors from the managed recovery process and that the recovery has applied the archived redo logs.

```
SELECT MAX(SEQUENCE#), THREAD# FROM V$LOG_HISTORY GROUP BY THREAD;
SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, CLIENT_PROCESS
    FROM V$MANAGED_STANDBY;
```

For a logical standby database, verify that there are no errors from the logical standby process and that the recovery has applied the archived redo logs.

```
SELECT THREAD#, SEQUENCE# SEQ#
    FROM DBA_LOGSTDBY_LOG LOG, DBA_LOGSTDBY_PROGRESS PROG
    WHERE PROG.APPLIED_SCN BETWEEN LOG.FIRST_CHANGE# AND LOG.NEXT_CHANGE#
    ORDER BY NEXT_CHANGE#;
```

Step 5: Restore Production Database Protection Mode

If you had to change the protection mode of the production database from maximum protection to either maximum availability or maximum performance because of the standby database outage, then change the production database protection mode back to maximum protection depending on your business requirements.

```
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE [PROTECTION | AVAILABILITY];
```

See Also: "Changing the Data Protection Mode" on page 7-24

Restoring Fault Tolerance after a Standby Database Data Failure

Following an unplanned outage of the standby database that requires a full or partial datafile restoration (such as data or media failure), full fault tolerance is compromised until the standby database is brought back into service. Full database protection should be restored as soon as possible. Note that using a Hardware Assisted Resilient Database configuration can prevent this type of problem.

> **See Also:** Appendix A, "Hardware Assisted Resilient Data (HARD) Initiative"

The following steps are required to restore full fault tolerance after data failure of the standby database:

- Step 1: Fix the Cause of the Outage
- Step 2: Restore the Backup of Affected Datafiles
- Step 3: Restore Required Archived Redo Log Files
- Step 5: Start Recovery or Apply
- Step 6: Verify Log Transport Services On the Production Database
- Step 7: Verify that Recovery or Apply Is Progressing On the Standby Database
- Step 8: Restore Production Database Protection Mode

Step 1: Fix the Cause of the Outage

The root cause of the outage should be investigated and action taken to prevent the problem from occurring again.

Step 2: Restore the Backup of Affected Datafiles

Only the affected datafiles need to be restored on to the standby site.

Step 3: Restore Required Archived Redo Log Files

Archived redo log files may need to be restored to recover the restored data files up to the configured lag.

For physical standby databases:

- If the archived redo logs required for recovery are available on the standby system in a configured archive destination, then the managed recovery process automatically finds and applies them as needed. No restoration is necessary.
- If the required archived redo logs have been deleted from the standby system but are still available on the production system, then the fetch archive log (FAL) process is automatically invoked to transfer them to the standby system. No restoration is necessary.

For logical standby databases, initiate complete media recovery for the affected files. Consider the following:

- If the archived redo logs required for recovery are available on the standby system in a configured archive destination, then the recovery process automatically finds and applies them as needed. No restoration is necessary.
- If the required archived redo logs have been deleted from the standby system, then they must be restored to the standby system. Complete media recovery for the affected files after the necessary archived redo logs are available.

Step 4: Start the Standby Database

After the standby database has been reestablished, start the standby database.

Type of Standby Database	SQL Statement
Physical	`STARTUP MOUNT;`
Logical	`STARTUP;`

Step 5: Start Recovery or Apply

Type of Standby Database	SQL Statement
Physical	`RECOVER MANAGED STANDBY DATABASE DISCONNECT;`
Logical	`ALTER DATABASE START LOGICAL STANDBY APPLY;`

Step 6: Verify Log Transport Services On the Production Database

Verify log transport services on the new production database by checking for errors when querying V$ARCHIVE_DEST and V$ARCHIVE_DEST_STATUS.

```
SELECT STATUS, TARGET, LOG_SEQUENCE, TYPE, PROCESS, REGISTER, ERROR FROM
V$ARCHIVE_DEST;
```

```
SELECT * FROM V$ARCHIVE_DEST_STATUS WHERE STATUS != 'INACTIVE';
```

Step 7: Verify that Recovery or Apply Is Progressing On the Standby Database

For a physical standby database, verify that there are no errors from the managed recovery process and that the recovery has applied archived redo logs.

```
SELECT MAX(SEQUENCE#), THREAD# FROM V$LOG_HISTORY GROUP BY THREAD;
SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, CLIENT_PROCESS
    FROM V$MANAGED_STANDBY;
```

For a logical standby database, verify that there are no errors from the logical standby process and that the recovery has applied archived redo logs.

```
SELECT THREAD#, SEQUENCE# SEQ#
    FROM DBA_LOGSTDBY_LOG LOG, DBALOGSTDBY_PROGRESS PROG
    WHERE PROG.APPLIED_SCN BETWEEN LOG.FIRST_CHANGE# AND LOG.NEXT_CHANGE#
    ORDER BY NEXT_CHANGE#;
```

Step 8: Restore Production Database Protection Mode

If you had to change the protection mode of the production database from maximum protection to either maximum availability or maximum performance because of the standby database outage, then change the production database protection mode back to maximum protection depending on your business requirements.

```
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE [PROTECTION | AVAILABILITY];
```

Restoring Fault Tolerance After the Production Database Has Opened Resetlogs

If the production database is activated because it was flashed back to correct a logical error or because it was restored and recovered to a point in time, then the corresponding standby database may require additional maintenance. No additional work is required if the production database did complete recovery with no resetlogs.

After activating the production database, execute the queries in the following table.

Database	Query
Production database	`SELECT TO_CHAR(RESETLOGS_CHANGE# - 2) FROM V$DATABASE;`
Physical standby database	`SELECT TO_CHAR(CURRENT_SCN) FROM V$DATABASE;`
Logical standby database	`SELECT APPLIED_SCN FROM DBA_LOGSTDBY_PROGRESS;`

Scenario 1: SCN on Standby is Behind Resetlogs SCN on Production

Database	Action
Physical standby database	1. Ensure that the standby database has received a new archived redo log file from the production database. 2. Restart recovery.
Logical standby database	Ensure that the standby database has received a new archived redo log file from the production database. `ALTER SYSTEM ARCHIVE_LOG_ CURRENT;`

Scenario 2: SCN on Standby is Ahead of Resetlogs SCN on Production

Database	Action

Physical standby database	1. Ensure that the standby database has received a new archived redo log file from the production database.
	2. Flash back the database to the SCN that is 2 SCNs before the resetlogs occurred.
	`SHUTDOWN IMMEDIATE; /* if necessary */`
	`STARTUP MOUNT;`
	`FLASHBACK DATABASE TO SCN` *`resetlogs_change#_minus_2`*`;`
	3. Restart recovery. See "Step 5P: Start Managed Recovery" on page 11-12.

Database	Action

Logical standby database	**1.** Retrieve production database flashback time or SCN. The flashback time or SCN needs to be extracted from the production database alert log.
	2. Stop SQL Apply on the logical standby database.
	`ALTER DATABASE STOP LOGICAL STANDBY APPLY;`
	`SELECT APPLIED_SCN FROM DBA_LOGSTDBY_PROGRESS;`
	3. Flash back the logical standby database.
	Issue the following SQL statements to flash back the logical standby database to the same time used to flash back the primary database.
	`SHUTDOWN;`
	`STARTUP MOUNT EXCLUSIVE;`
	`FLASHBACK DATABASE TO TIMESTAMP` *`time_of_primary_database_flashback`*`;`
	`ALTER DATABASE OPEN READ ONLY;`
	`SELECT APPLIED_SCN FROM DBA_LOGSTDBY_PROGRESS;`
	4. Open the logical standby database with resetlogs
	`SHUTDOWN;`
	`STARTUP MOUNT EXCLUSIVE;`
	`ALTER DATABASE OPEN RESETLOGS;`
	5. Archive the current log on the primary database.
	`ALTER SYSTEM ARCHIVE LOG CURRENT;`
	6. Start SQL Apply.
	`ALTER DATABASE START LOGICAL STANDBY APPLY;`

Restoring Fault Tolerance after Dual Failures

If a dual failure affecting both the standby and production databases occurs, then you need to re-create the production database first. Because the sites are identical, the production database can be created wherever the most recent backup resides.

Table 11–3 summarizes the recovery strategy depending on the type of backups that are available.

Table 11–3 Re-Creating the Production and Standby Databases

Available Backups	Re-Creating the Production Database
Local backup on production and standby databases	Restore backup from the production database. Recover and activate the database as the new production database.
Local backup only on standby database. Tape backups on standby database.	Restore the local standby backup to the standby database. Recover and activate the database as the new production database.
Tape backups only	Restore tape backups locally. Recover the database and activate it as the new production database.

After the production database is re-created, follow the steps for creating a new standby database that are described in *Oracle Data Guard Concepts and Administration*.

A

Hardware Assisted Resilient Data (HARD) Initiative

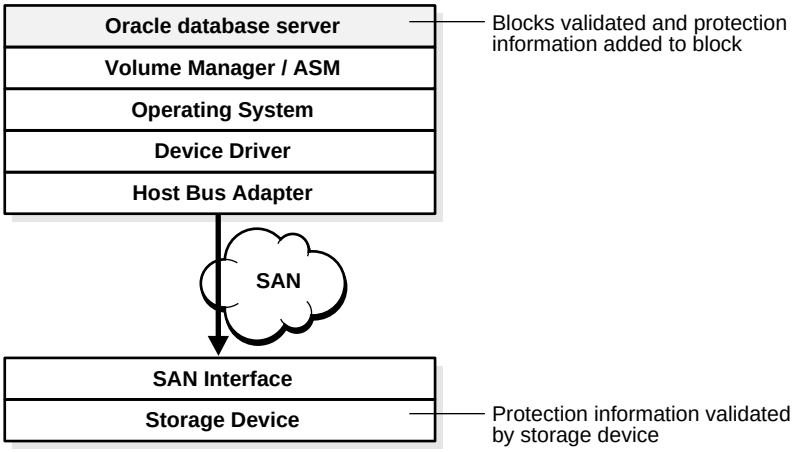
Preventing Data Corruptions with HARD-Compliant Storage

Oracle has introduced the Hardware Assisted Resilient Data (HARD) Initiative, which is a program designed to prevent data corruptions before they happen. Data corruptions are very rare, but when they happen, they can have a catastrophic effect on a database, and therefore a business.

Under the HARD Initiative, Oracle continues to work with selected system and storage vendors to build operating system and storage components that can detect corruptions early and prevent corrupted data from being written to disk. The kay approach is block checking where the storage subsystem validates the Oracle block contents. Implementation of this feature is transparent to the end user or DBA, regardless of the hardware vendor.

To use HARD validation, all datafiles and log files are placed on HARD-compliant storage. The user must also enable the HARD validation feature on the storage, using the vendor-provided interface. When Oracle writes data to the storage, the storage system validates the data. If it appears to be corrupted, then the write is rejected with an error.

Figure A–1 Oracle Data Validation



Data Corruptions

Data corruption may occur due to many reasons: a bit-flip on a disk; a software bug in a database, an operating system, a storage area network (SAN), or a storage system. If not prevented or repaired, data corruptions can bring down a database or cause key business data to be lost.

Oracle provides sophisticated techniques for detecting data corruptions and recoverying from them. These techniques include block-level recovery, automated backup and restore, tablespace point-in-time recovery, remote standby databases, and transactional recovery. However, recovering from a corruption can take a long time. Furthermore, corruptions to critical data can cause the entire database to fail. It is better to prevent the data corruption in the first place. HARD provides the mechanism that prevents business data from becoming corrupted

Oracle's Maximum Availability Architecture (MAA) describes an infrastructure that reduces the impact of outages using Oracle RAC, Oracle Data Guard, and HARD.

Types of Data Corruption Addressed by HARD

The HARD program defines multiple levels of protection. Oracle storage partners may choose to implement firmware or hardware checks that can prevent the following types of corruptions:

- Writes of corrupted blocks

Data that is corrupted by some intervening operating system or hardware component after the data was written by Oracle and before it reaches the disk. These components might include the operating system, file system, the volume manager, device driver, the host bus adapter, and the SAN switching fabric. Though Oracle can detect the corruption when reading the data back, Oracle may not read the data until days or months later. By then, a good backup for recovering the data may no longer be available.

- Writes of blocks to incorrect locations

  Oracle issues a write to a specific location on disk. Somehow the operating system or storage system writes the blocks to the wrong location. This can cause two corruptions: corrupting valid data on the disk and losing the data from the committed transaction.

- Erroneous writes by programs other than Oracle to Oracle data

  Oracle datafiles might be overwritten by non-Oracle applications. A non-Oracle process or program may accidentally overwrite the contents of an Oracle datafile. This can be either because of a bug in the application software, operating system, or human error (for example, accidentally copying a normal operating system file over an Oracle datafile).

- Corrupted third-party backups

  Data corruptions can occur when backups are copied to tape. This type of corruption is particularly pernicious because the backups are used to repair data corruptions. So if the backups are also corrupted, then there is no way to recover any lost data. This is especially true for third-party backups (in which the disk storage unit directly copies data to the backup device without going through Oracle.)

Possible HARD Checks

In a storage system where the Oracle HARD functionality is implemented, the Oracle server can validate the Oracle block structure, block integrity, and block location with numerous checks. If a block fails validation when it is written, then the storage rejects the write and thereby protects the integrity of the data. The HARD validation checks can also be selectively disabled during system management operations that may temporarily leave data in an inconsistent state.

The following Oracle objects can be validated during I/O writes:

- Datafile

- Control file
- Redo log
- Archived redo log
- RMAN backup piece
- Flashback log
- Change tracking file
- ASM metadata
- Oracle Cluster Registry file
- Data Guard Broker configuration file

DBAs and system administrators using HARD should be aware that a HARD error will be reported back through to the ORACLE instance as an I/O error. System administrators must carefully check the system log to check for HARD-enabled storage before starting any recovery actions.

Automatic storage management (ASM) rebalancing is currently not supported with HARD implementations. ASM rebalancing moves the complete disk block, which may be larger than the datablock and have spurious characters not covered by HARD checking. This would cause a false HARD validation error.

Storage vendors may choose to implement some or all of the checks in their implementation. Also, each vendor's implementation is unique and their control interfaces may have different features. Please check with the HARD Initiative page for the latest vendor and implementation information.

`http://otn.oracle.com/deploy/availability/index.html`

B

Database SPFILE and Oracle Net Configuration File Samples

The tables and file samples in this appendix are included to illustrate the best practices as they relate to different HA architectures. These samples also clarify how the database server parameter file (SPFILE) relates to the Oracle Net configuration for dynamic service registration.

The following tables and sample files are included in this appendix:

- SPFILE Samples
  - Table B–1, "Generic SPFILE Parameters for Primary, Physical Standby, and Logical Standby Databases"
  - Table B–2, "RAC SPFILE Parameters for Primary, Physical Standby, and Logical Standby Databases"
  - Table B–3, "Data Guard SPFILE Parameters for Primary Database and Physical Standby Database Only"
  - Table B–4, "Data Guard SPFILE Parameters for Primary Database and Logical Standby Database Only"
  - Table B–5, "Data Guard SPFILE Parameters for Primary Database, Physical Standby Database, and Logical Standby Database"
  - Table B–6, "Data Guard SPFILE Parameters for Maximum Performance Mode"
- Oracle Net Configuration Files
  - SQLNET.ORA File Example for All Hosts Using Dynamic Instance Registration

- LISTENER.ORA File Example for All Hosts Using Dynamic Instance Registration
- TNSNAMES.ORA File Example for All Hosts Using Dynamic Instance Registration

The tables and files are shown for the following configuration:

- ORACLE_BASE=/mnt/app/oracle
- Database flash recovery area is /flash_recovery

SPFILE Samples

The tables in this section represent the database, RAC, and Data Guard parameter file values. Some parameters appear in both the generic database parameter table and the RAC parameter table. If RAC is being used, then the value in the RAC parameter table should be used instead of the value in the generic database parameter table.

The parameters show the configuration for a database in Chicago and an option for a physical standby database and a logical standby database in Boston. The primary database is the SALES database. For a single instance database, the ORACLE_SID parameter values are SALES, SALES_PHYS, and SALES_LOG. In a RAC configuration, the corresponding instance number is appended to each of the ORACLE_SID parameter values.

Table B–1 Generic SPFILE Parameters for Primary, Physical Standby, and Logical Standby Databases

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.COMPATIBLE='10.1.0.1.0'	Same as Chicago	Same as Chicago
*.LOG_ARCHIVE_FORMAT='arch_%t_%S_%r.log'	Same as Chicago	Same as Chicago
*.LOG_ARCHIVE_TRACE=0	Same as Chicago	Same as Chicago
*.REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE	Same as Chicago	Same as Chicago
*.LOG_CHECKPOINT_INTERVAL=0	Same as Chicago	Same as Chicago
*.LOG_CHECKPOINT_TIMEOUT=0	Same as Chicago	Same as Chicago
*.LOG_CHECKPOINTS_TO_ALERT=TRUE	Same as Chicago	Same as Chicago

Table B–1 Generic SPFILE Parameters for Primary, Physical Standby, and Logical Standby Databases

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.DB_BLOCK_CHECKING=TRUE	Same as Chicago	Same as Chicago
*.DB_BLOCK_CHECKSUM=TRUE	Same as Chicago	Same as Chicago
*.TIMED_STATISTICS=TRUE	Same as Chicago	Same as Chicago
*.LOCAL_LISTENER='SALES_lsnr'	Same as Chicago	Same as Chicago
*.REMOTE_LISTENER='SALES_remotelsnr_CHICAGO'	*.REMOTE_LISTENER='SALES_remotelsnr_BOSTON'	*.REMOTE_LISTENER='SALES_remotelsnr_BOSTON'
*.DB_RECOVERY_FILE_DEST=/flash_recovery	Same as Chicago	Same as Chicago
*.DB_RECOVERY_FILE_SIZE=100G	Same as Chicago	Same as Chicago
*.DB_FLASHBACK_RETENTION_TARGET=240	Same as Chicago	Same as Chicago
*.UNDO_MANAGEMENT=AUTO	Same as Chicago	Same as Chicago
*.UNDO_RETENTION=900	Same as Chicago	Same as Chicago
*.UNDO_TABLESPACE='rbs01'	Same as Chicago	Same as Chicago
*.DB_NAME='SALES'	Same as Chicago	*.DB_NAME='SALES_LOG'
*.SERVICE_NAME='SALES_CHICAGO'	*.SERVICE_NAME='SALES_BOSTON'	*.SERVICE_NAME='SALES_BOSTON'
*.BACKGROUND_DUMP_DEST='mnt/app/oracle/admin/SALES/bdump'	*.BACKGROUND_DUMP_DEST='mnt/app/oracle/admin/SALES/bdump'	*.BACKGROUND_DUMP_DEST='mnt/app/oracle/admin/SALES_LOG/bdump'
*.CORE_DUMP_DEST='/mnt/app/oracle/admin/SALES/cdump'	*.CORE_DUMP_DEST='/mnt/app/oracle/admin/SALES/cdump'	*.CORE_DUMP_DEST='/mnt/app/oracle/admin/SALES_LOG/cdump'
*.USER_DUMP_DEST='/mnt/app/oracle/admin/SALES/udump'	*.USER_DUMP_DEST='/mnt/app/oracle/admin/SALES/udump'	*.USER_DUMP_DEST='/mnt/app/oracle/admin/SALES_LOG/udump'
*.CLUSTER_DATABASE=FALSE	Same as Chicago	Same as Chicago
*.CONTROL_FILES='/oradata/SALES/SALES_cntr01','/oradata/SALES/SALES_cntr02'	*.CONTROL_FILES='/oradata/SALES/SALES_cntr01','/oradata/SALES/SALES_cntr02'	*.CONTROL_FILES='/oradata/SALES_LOG/SALES_cntr01','/oradata/SALES_LOG/SALES_cntr02'

Table B–1 Generic SPFILE Parameters for Primary, Physical Standby, and Logical Standby Databases

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.DB_FILE_NAME_ CONVERT='/SALES_ LOG/','/SALES/'	Same as Chicago	*.DB_FILE_NAME_ CONVERT='/SALES/','/SALES_ LOG/'
*.LOG_FILE_NAME_ CONVERT='/SALES_ LOG/','/SALES/'	Same as Chicago	*.LOG_FILE_NAME_ CONVERT='/SALES/','/SALES_ LOG/'
*.STANDBY_FILE_ MANAGEMENT=AUTO	Same as Chicago	Same as Chicago
*.CONTROL_FILE_RECORD_ KEEP_TIME=30	Same as Chicago	Same as Chicago
*.RESUMABLE_TIMEOUT=900	Same as Chicago	Same as Chicago
*.INSTANCE_NAME=SALES_ CHICAGO	*INSTANCE_NAME=SALES_ BOSTON	INSTANCE_NAME=SALES_ BOSTON_LOG

Table B–2 RAC SPFILE Parameters for Primary, Physical Standby, and Logical Standby Databases

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.CLUSTER_DATABASE=TRUE	Same as Chicago	Same as Chicago
SALES1.THREAD=1	SALES_PHYS1.THREAD=1	SALES_LOG1.THREAD=1
SALES2.THREAD=2	SALES_PHYS2.THREAD=2	SALES_LOG2.THREAD=2
SALES1.INSTANCE_NUMBER=1	SALES_PHYS1.INSTANCE_ NUMBER=1	SALES_LOG1.INSTANCE_ NUMBER=1
SALES2.INSTANCE_NUMBER=2	SALES_PHYS2.INSTANCE_ NUMBER=2	SALES_LOG2.INSTANCE_ NUMBER=2
SALES1.INSTANCE_ NAME=SALES_CHICAGO1	SALES_PHYS1.INSTANCE_ NAME=SALES_BOSTON1	SALES_LOG1.INSTANCE_ NAME=SALES_BOSTON1
SALES2.INSTANCE_ NAME=SALES_CHICAGO2	SALES_PHYS2.INSTANCE_ NAME=SALES_BOSTON2	SALES_LOG2.INSTANCE_ NAME=SALES_BOSTON2
SALES1.UNDO_ TABLESPACE='rbs01'	SALES_PHYS1.UNDO_ TABLESPACE='rbs01'	SALES_LOG1.UNDO_ TABLESPACE='rbs01'
SALES2.UNDO_ TABLESPACE='rbs02'	SALES_PHYS2.UNDO_ TABLESPACE='rbs02'	SALES_LOG2.UNDO_ TABLESPACE='rbs02'

Table B–2 RAC SPFILE Parameters for Primary, Physical Standby, and Logical Standby Databases

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.STANDBY_FILE_ MANAGEMENT=MANUAL	Same as Chicago	Same as Chicago

Table B–3 Data Guard SPFILE Parameters for Primary Database and Physical Standby Database Only

Chicago (Primary Database)	Boston (Physical Standby Database)
*.FAL_CLIENT='SALES_CHICAGO'	*FAL_CLIENT='SALES_BOSTON'
*.FAL_SERVER='SALES_BOSTON'	*FAL_SERVER='SALES_CHICAGO'
*.DB_UNIQUE_NAME-'SALES_CHICAGO'	*.DB_UNIQUE_NAME-'SALES_BOSTON'
*.LOG_ARCHIVE_CONFIG='DG_CONFIG=(SALES_ CHICAGO,SALES_BOSTON)'	Same as Chicago
*STANDBY_ARCHIVE_DEST=USE_DB_RECOVERY_ FILE_DEST	Same as Chicago
*.LOG_ARCHIVE_DEST_1='location=USE_DB_ RECOVERY_FILE_DEST arch noreopen max_ failure=0 mandatory valid_for=(ALL_ LOGFILES,ALL_ROLES) db_unique_ name=SALES_CHICAGO'	*.LOG_ARCHIVE_DEST_1='location=USE_DB_ RECOVERY_FILE_DEST arch noreopen max_ failure=0 mandatory valid_for=(ALL_ LOGFILES,ALL_ROLES) db_unique_ name=SALES_BOSTON'
*.LOG_ARCHIVE_DEST_2='service=SALES_ BOSTON reopen=15 max_failure=10 lgwr sync=noparallel affirm valid_ for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_ unique_name=SALES_BOSTON'	*.LOG_ARCHIVE_DEST_2='service=SALES_ BOSTON reopen=15 max_failure=10 lgwr sync=noparallel affirm valid_ for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_ unique_name=SALES_CHICAGO'

Table B–4 Data Guard SPFILE Parameters for Primary Database and Logical Standby Database Only

Chicago (Primary Database)	Boston (Logical Standby Database)
*.FAL_CLIENT='SALES_CHICAGO'	*.FAL_CLIENT='SALES_BOSTON_LOG'
*.FAL_SERVER='SALES_BOSTON_LOG'	*.FAL_SERVER='SALES_CHICAGO'
*.DB_UNIQUE_NAME=SALES_CHICAGO'	*.DB_UNIQUE_NAME='SALES_BOSTON_LOG'
*.LOG_ARCHIVE_CONFIG='DG_CONFIG=(SALES_ CHICAGO,SALES_BOSTON_LOG)'	Same as Chicago
*.STANDBY_ARCHIVE_ DEST='/arch/SALES/archivelog'	*.STANDBY_ARCHIVE_DEST='/arch/SALES_ LOG/archivelog'

Table B–4 Data Guard SPFILE Parameters for Primary Database and Logical Standby Database Only

Chicago (Primary Database)	Boston (Logical Standby Database)
*.LOG_ARCHIVE_DEST_1='location=USE_DB_ RECOVERY_FILE_DEST arch noreopen max_ failure=0 mandatory valid_for=(ONLINE_ LOGFILES,ALL_ROLES) db_unique_ name=SALES_CHICAGO'	*.LOG_ARCHIVE_DEST_1='location=USE_DB_ RECOVERY_FILE_DEST arch noreopen max_ failure=0 mandatory valid_for=(ONLINE_ LOGFILES,ALL_ROLES) db_unique_ name=SALES_BOSTON_LOG'
*.LOG_ARCHIVE_DEST_3='service=SALES_ BOSTON_LOG reopen=15 max_failure=10 lgwr sync=noparallel affirm valid_ for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_ unique_name=SALES_BOSTON_LOG'	*.LOG_ARCHIVE_DEST_3='service=SALES_ CHICAGO reopen=15 max_failure=10 lgwr sync=noparallel affirm valid_ for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_ unique_name=SALES_CHICAGO'
*.LOG_ARCHIVE_DEST_ 4='location=/arch/SALES/archivelog arch noreopen max_failure=0 mandatory valid_ for=(STANDBY_LOGFILES,ALL_ROLES) db_ unique_name=SALES_CHICAGO'	*.LOG_ARCHIVE_DEST_ 4='location=/arch/SALES_LOG/archivelog arch noreopen max_failure=0 mandatory valid_for=(STANDBY_LOGFILES,ALL_ROLES) db_unique_name=SALES_BOSTON_LOG'
*.PARALLEL_MAX_SERVERS=9	Same as Chicago

Table B–5 applies to a Data Guard environment running in either maximum availability mode or maximum protection mode.

Table B–5 Data Guard SPFILE Parameters for Primary Database, Physical Standby Database, and Logical Standby Database

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.FAL_CLIENT='SALES_ CHICAGO'	*.FAL_CLIENT='SALES_ BOSTON'	*.FAL_CLIENT='SALES_ BOSTON_LOG'
*.FAL_SERVER='SALES_ BOSTON','SALES_BOSTON_LOG'	*.FAL_SERVER='SALES_ BOSTON','SALES_BOSTON_LOG'	*.FAL_SERVER='SALES_ BOSTON','SALES_BOSTON'
*.DB_UNIQUE_NAME='SALES_ CHICAGO'	*.DB_UNIQUE_NAME='SALES_ BOSTON'	*.DB_UNIQUE_NAME='SALES_ BOSTON_LOG'
*.LOG_ARCHIVE_CONFIG='DG_ CONFIG=(SALES_ CHICAGO,SALES_ BOSTON,SALES_BOSTON_LOG)'	Same as Chicago	Same as Chicago
*.STANDBY_ARCHIVE_ DEST='/arch/SALES/archivel og'	Same as Chicago	*.STANDBY_ARCHIVE_ DEST='/arch/SALES_ LOG/archivelog

Table B–5 Data Guard SPFILE Parameters for Primary Database, Physical Standby Database, and Logical Standby Database (Cont.)

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.LOG_ARCHIVE_DEST_ 1='location=USE_DB_ RECOVERY_FILE_DEST arch noreopen max_failure=0 mandatory valid_ for=(ONLINE_LOGFILES,ALL_ ROLES) db_unique_ name=SALES_CHICAGO'	*.LOG_ARCHIVE_DEST_ 1='location=USE_DB_ RECOVERY_FILE_DEST arch noreopen max_failure=0 mandatory valid_ for=(ONLINE_LOGFILES,ALL_ ROLES) db_unique_ name=SALES_BOSTON'	*.LOG_ARCHIVE_DEST_ 1='location=USE_DB_ RECOVERY_FILE_DEST arch noreopen max_failure=0 mandatory valid_ for=(ONLINE_LOGFILES,ALL_ ROLES) db_unique_ name=SALES_BOSTON_LOG'
*.LOG_ARCHIVE_DEST_ 2='service=SALES_BOSTON reopen=15 max_failure=10 lgwr sync=noparallel affirm valid_for=(ONLINE_ LOGFILES,ALL_ROLES) db_ unique_name=SALES_BOSTON'	*.LOG_ARCHIVE_DEST_ 2='service=SALES_CHICAGO reopen=15 max_failure=10 lgwr sync=noparallel affirm valid_for=(ONLINE_ LOGFILES,ALL_ROLES) db_ unique_name=SALES_CHICAGO'	N / A
*.LOG_ARCHIVE_DEST_ 3='service=SALES_BOSTON_ LOG reopen=15 max_ failure=10 lgwr sync=noparallel affirm valid_for=(ONLINE_ LOGFILES,PRIMARY_ROLE) db_ unique_name=SALES_BOSTON_ LOG'	*.LOG_ARCHIVE_DEST_ 3='service=SALES_BOSTON_ LOG reopen=15 max_ failure=10 lgwr sync=noparallel affirm valid_for=(ONLINE_ LOGFILES,PRIMARY_ROLE) db_ unique_name=SALES_BOSTON_ LOG'	*.LOG_ARCHIVE_DEST_ 3='service=SALES_CHICAGO reopen=15 max_failure=10 lgwr sync=noparallel affirm valid_for=(ONLINE_ LOGFILES,PRIMARY_ROLE) db_ unique_name=SALES_CHICAGO'
*.LOG_ARCHIVE_DEST_ 4='location=/arch/SALES/ar chivelog arch noreopen max_failure=0 mandatory valid_for=(STANDBY_ LOGFILES,ALL_ROLES) db_ unique_name=SALES_CHICAGO'	*.LOG_ARCHIVE_DEST_ 4='location=/arch/SALES/ar chivelog arch noreopen max_failure=0 mandatory valid_for=(STANDBY_ LOGFILES,ALL_ROLES) db_ unique_name=SALES_BOSTON'	*.LOG_ARCHIVE_DEST_ 4='location=/arch/SALES_ LOG/archivelog arch noreopen max_failure=0 mandatory valid_ for=(STANDBY_LOGFILES,ALL_ ROLES) db_unique_ name=SALES_BOSTON_LOG'
*.PARALLEL_MAX_SERVERS=9	Same as Chicago	Same as Chicago

Table B–6 shows how to change the parameters for a Data Guard environment that is running in maximum performance mode.

Table B–6 Data Guard SPFILE Parameters for Maximum Performance Mode

Chicago (Primary Database)	Boston (Physical Standby Database)	Boston (Logical Standby Database)
*.LOG_ARCHIVE_DEST_2='service=SALES_BOSTON reopen=15 max_failure=10 lgwr async=102400 net_timeout=30 valid_for=(ONLINE_LOGFILES,ALL_ROLES) db_unique_name=SALES_BOSTON'	*.LOG_ARCHIVE_DEST_2='service=SALES_CHICAGO reopen=15 max_failure=10 lgwr async=102400 net_timeout=30 valid_for=(ONLINE_LOGFILES,ALL_ROLES) db_unique_name=SALES_CHICAGO'	N/A
*.LOG_ARCHIVE_DEST_3='service=SALES_BOSTON_LOG reopen=15 max_failure=10 lgwr async=102400 net_timeout=30 valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_unique_name=SALES_BOSTON_LOG'	*.LOG_ARCHIVE_DEST_3='service=SALES_BOSTON_LOG reopen=15 max_failure=10 lgwr async=102400 net_timeout=30 valid_for=(ONLINE_LOGFILES,ALL_ROLES) db_unique_name=SALES_BOSTON_LOG'	*.LOG_ARCHIVE_DEST_3='service=SALES_CHICAGO reopen=15 max_failure=10 lgwr async=102400 net_timeout=30 valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_unique_name=SALES_CHICAGO'

Oracle Net Configuration Files

SQLNET.ORA File Example for All Hosts Using Dynamic Instance Registration

```
# Set dead connection time
SQLNET.EXPIRE_TIME = 1
# Set default SDU for all connections
DEFAULT_SDU_SIZE=32767
```

LISTENER.ORA File Example for All Hosts Using Dynamic Instance Registration

For a RAC environment, listeners must be listening on the virtual IP addresses (VIP), rather than the local host name.

```
lsnr_SALES =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST=
        (ADDRESS=(PROTOCOL=tcp)(HOST=<local_host_name>)(PORT=1513)
                 (QUEUESIZE=1024)))))
# Password Protect listener; See "Oracle Net Services Administration Guide"
```

```
PASSWORDS_lsnr_SALES = 876EAE4513718ED9
# Prevent listener administration
ADMIN_RESTRICTIONS_lsnr_SALES=ON
```

TNSNAMES.ORA File Example for All Hosts Using Dynamic Instance Registration

```
# Used for database parameter local_listener
SALES_lsnr =
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(PORT=1513)))

SALES_remotelsnr_CHICAGO =
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<chicago_host1>))
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<chicago_host2>)))

SALES_remotelsnr_BOSTON =
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<boston_host1>))
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<boston_host2>)))


# Net service used for communication with SALES database in Chicago
SALES_CHICAGO =
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<chicago_host1>))
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<chicago_host2>)))
    (CONNECT_DATA=(SERVICE_NAME=SALES_CHICAGO)))


# Net service used for communication with SALES database in Boston
SALES_BOSTON =
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<boston_host1>))
      (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<boston_host2>)))
    (CONNECT_DATA=(SERVICE_NAME=SALES_BOSTON)))

# Net service used for communication with Logical Standby SALES database in
Boston
SALES_BOSTON_LOG =
  (DESCRIPTION=
    (ADDRESS_LIST=
```

```
    (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<boston_host1>))
    (ADDRESS=(PROTOCOL=tcp)(PORT=1513)(HOST=<boston_host2>)))
  (CONNECT_DATA=(SERVICE_NAME=SALES_BOSTON_LOG)))
```

Index

A

alerts
  Enterprise Manager, 8-3
application failover, 10-2
  configuration recommendations, 7-44
  RAC not deployed, 7-47
application service brownouts, 8-9
apply instance failover, 10-2, 10-18
  using SQL*Plus, 10-19
ARCH attribute of the LOG_ARCHIVE_DEST_n
    initialization parameter, 7-28
architecture
  Data Guard only, 4-2
  database only, 4-2
  MAA, 4-2
  Maximum Availability Architecture, 4-2
  RAC only, 4-2
  Streams, 4-2
archived redo log
  recovering, 10-32
ARCHIVELOG mode, 7-4
archiving strategy, 7-16
ASM, 6-5
  striping, 6-3
ASM disk group, 6-7
ASM failure groups, 6-6
ASYNC attribute, 7-29
async buffer size, 7-29
authentication checks, 7-31
automatic checkpoint tuning, 7-6
automatic segment space management, 7-9
automatic storage management, 6-5
  description, 3-6
  striping, 6-3
automatic undo management, 7-7
availability
  definition, 1-2

B

backup and recovery
  automatic disk-based, 7-40
  double failures, 7-38
  flash recovery area, 7-41
  Oracle Cluster Registry, 7-43
  recommendations, 7-36
  schedule, 7-38
backup and recovery plans, 5-7
backup retention policy, 7-40
BACKUP VALIDATE RMAN command, 7-43
backups
  incremental, 7-39
  long-term, 7-39
bandwidth, 7-25
BLOCK CHANGE TRACKING, 7-43
block checking, 7-5
block checksums, 7-4
block media recovery, 10-33
block validation, 6-8
boot disk, 6-10
brownouts, 8-9

C

capacity planning, 5-4
CFS (cluster filesystem), 6-5
change control, 5-5

change management, 5-4
change tracking, 7-40
checkpoints, 7-5
cluster filesystem, 6-5
cluster interconnect, 6-11
Cluster Ready Services, 7-44
  configuration recommendations, 7-50
CLUSTER_INTERCONNECTS initialization parameter, 7-13
clustering software, 6-13
clusterwide outage
  restoring, 11-14
cold failover, 4-15
connection descriptor
  parameters, 7-49
connection descriptors
  production instances, 7-45
connect-time failover, 7-36
control file
  recovering loss, 10-30
  RMAN repository, 7-42
control file copies, 7-2
CONTROL_FILE_RECORD_KEEP_TIME initialization parameter, 7-3
corruption
  BACKUP VALIDATE RMAN command, 7-43
  recovery, 10-22
CPUs
  recommended number, 6-9
CREATE DISKGROUP statement, 6-7
CRS
  configuration recommendations, 7-50

D

data corruption
  preventing, A-1
data failure
  manual re-creation, 10-34
  recovery, 10-2, 10-22
  recovery with Data Guard, 10-27
  recovery without Data Guard, 10-26
  RMAN block media recovery, 10-33
  RMAN datafile media recovery, 10-33
  using Data Guard to recover, 10-35
data failure on standby database
  restoring, 11-16
data failures
  file or block media recovery, 7-38
Data Guard
  benefits, 3-2
  choosing failover, 10-10
  configuration recommendations, 7-13
  connect-time failover, 7-36
  failover, 10-10
  monitoring with Enterprise Manager, 8-11
  recovering from data failure, 10-35
  switchover, 10-12
  using Enterprise Manager to manage targets, 8-12
Data Guard failover
  using SQL*Plus, 10-11
Data Guard only architecture, 4-2
  benefits, 4-7
Data Guard switchover
  choosing, 10-13
  using SQL*Plus, 10-13
data protection mode
  changing, 7-24
data protection modes, 7-22
  network configuration, 7-26
database area, 6-4
database configuration
  recommendations, 7-2
database failover, 10-10
  recovery, 10-2
database only architecture, 4-2
  HA features and descriptions, 4-3
database patch upgrades
  recommendations, 10-48
Database Resource Manager, 7-11
database switchover, 10-2, 10-12
datafile
  recovering, 10-30
datafile block corruption
  ANALYZE statement, 10-25
  DBMS_REPAIR package, 10-25
  DBVERIFY utility, 10-25
  detecting, 10-23
  recovery, 10-23

RMAN, 10-25
DB_FLASHBACK_RETENTION_TARGET initialization parameter, 7-11
DB_RECOVERY_FILE_DEST initialization parameter, 7-10
DB_RECOVERY_FILE_DEST_SIZE initialization parameter, 7-10
DB_UNIQUE_NAME initialization parameter, 7-32
DBMS_LOGSTDBY.SKIP procedure, 7-35
DBMS_REDEFINITION PL/SQL package, 10-54
DELAY parameter, 7-49
disaster recovery planning, 5-8
DNS failover, 10-7
downtime
causes, 1-4
cost, 2-3
driver version, 6-12
dropped tablespace
using Flashback Database, 10-44
dropping database objects, 10-42
dual failures
restoring, 11-20
dynamic reconfiguration
description, 3-8
dynamic service registration, 7-20

E

Enterprise Manager
alerts, 8-3
availability, 8-13
configuring listener, 8-19
Database Targets page, 8-9
HA architecture, 8-13
HA architecture recommendations, 8-15
location of Management Repository, 8-20
managing Data Guard targets, 8-12
managing metrics, 8-10
managing patches, 8-12
metric, 8-4
monitoring Data Guard, 8-11
Notification Rules, 8-5
performance, 8-9
Policy Violations, 8-12
recommended notification rules, 8-10
unscheduled outages of Enterprise Manager, 8-17
EXTERNAL REDUNDANCY clause, 6-7

F

failover
apply instance, 10-18
apply instance using SQL*Plus, 10-19
Data Guard, 10-10
database, 10-10
RAC and Data Guard, 10-18
FAILOVER parameter, 7-49
failure detection
operating system, 6-12
failure groups, 6-6
FAL_SERVER and FAL_CLIENT initialization parameters, 7-20
FAST_START_MTTR_TARGET initialization parameter, 7-6
fast-start checkpointing, 7-5
fencing
operation system, 6-12
flash recovery area, 6-5
backups, 7-41
description, 3-11
recommendations for configuring, 7-10
size, 7-41
tape backups, 7-40
Flashback Database, 10-37, 10-43
description, 3-8
enabling, 7-10
Flashback Drop, 10-37, 10-42
description, 3-8
Flashback Query, 10-36, 10-38
description, 3-7
Flashback Table, 10-37, 10-41
description, 3-8
flashback technology
example, 10-39
recovering from user error, 10-35
Flashback Transaction Query, 10-37, 10-39
description, 3-8
Flashback Version Query, 10-36, 10-38

description, 3-8
FORCE LOGGING mode, 7-19

G

Grid Control home page, 8-3

H

HA architectures
comparison, 4-12
HARD initiative, 3-11, 6-8, A-1
hardware
fencing, 6-10
Hardware Assisted Resilient Data (HARD) initiative, 3-11, A-1
Hardware Assisted Resilient Data initiative, 6-8
hardware components
redundant, 6-3
high availability
business impact analysis, 2-3
importance, 1-3
importance of documentation, 5-11
operational policies, 5-2
training people, 5-11
high availability architecture
characteristics, 1-3
high availability solution
characteristics, 1-2

I

identical hardware, 6-11
index rebuilding, 10-54
initialization parameters
primary and physical standby example, 7-17
intelligent storage arrays, 6-7
interim patch upgrade, 10-45
I/O operations
load balancing, 6-4

J

JDBC fast connection failover, 7-44, 7-47
journal file systems, 6-13

L

latency
effect on primary throughput, 7-28
listener.ora file, 7-21
listener.ora file sample, N-8
load balancers
application server, 6-16
network, 6-16
load balancing
I/O operations, 6-4
LOAD_BALANCE parameter, 7-49
local archiving first, 7-17
locally managed tablespaces, 7-9
LOG_ARCHIVE_DEST_2 initialization parameter, 7-20
LOG_ARCHIVE_FORMAT initialization parameter, 7-17
LOG_ARCHIVE_LOCAL_FIRST initializatioin parameter, 7-17
LOG_ARCHIVE_LOCAL_FIRST initialization parameter, 7-31
logging, 6-13
logical standby archive destination, 7-17
logical standby database
configuration recommendations, 7-14, 7-33
restoring, 11-13
upgrade, 10-49
upgrades, 10-3
logical standby failover
using SQL*Plus, 10-12
logical standby switchover
using SQL*Plus, 10-14
logical volume, 6-5

M

MAA, 4-2
benefits, 4-9
configuration recommendations, 7-35
Management Agent, 8-2
MAX_SERVERS initialization parameter, 7-33
Maximum Availability Architecture, 4-2
benefits, 4-9
maximum availability mode, 7-23
maximum performance mode, 7-23

maximum protection mode, 7-22
media failure
  datafile recovery, 10-30
  recovery, 10-22, 10-29
media recovery
  performance, 7-32
metric
  Enterprise Manager, 8-4
middle-tier applications
  service callouts for notification, 7-50
mirroring and striping data, 6-3
mirroring disks, 6-13
monitoring
  Enterprise Manager, 8-2
multiple node failures, 10-16
multiple standby instances, 7-35

N

network components
  redundant, 6-14
network configuration
  performance assessment, 7-25
network recommendations
  all architectures, 6-14
  RAC, 6-17
network recovery objective, 2-4
network time protocol, 6-14
nod failure
  multiple, 10-16
node failure
  single, 10-16
notification
  RAC not deployed, 7-47
notification rules
  recommended, 8-10
notifications
  middle-tier applications, 7-50
NRO, 2-4
NTP, 6-14

O

object reorganization, 10-3
OCR
  backing up, 7-43
  recovering, 10-32
OCR protection, 6-9
ocrconfig tool
  backing up Oracle Cluster Registry, 7-43
OIFCFG, 6-17
online index reorganization, 10-54
online object reorganization, 10-3, 10-53
online redo log file
  recovering, 10-31
online redo log files
  multiplex, 7-3
online reorganization
  description, 3-5
online servicing, 6-3
online table reorganization, 10-54
online tablespace reorganization, 10-54
opatch
  applying a patch, 10-46
  listing installed software and patches, 10-47
  rolling back a patch, 10-47
opatch command-line utility, 10-45
operating system parameters, 6-13
operating system version, 6-12
Oracle Advanced Security, 7-31
Oracle Cluster Registry
  backing up, 7-43
  backups, 7-43
  recovering, 10-32
Oracle Cluster Registry protection, 6-9
Oracle Fail Safe
  description, 3-9
Oracle Interface Configuration, 6-17
Oracle Net configuration files
  samples, N-8
outages
  scheduled, 9-8
  unscheduled, 9-2

P

PARALLEL_MAX_SERVERS initialization
    parameter, 7-34
partial site failover
  network routes, 10-8

patch
applying with opatch, 10-46
rolling back, 10-47
patch level, 6-12
patch upgrade, 10-3, 10-45
patch upgrades
rolling, 10-45
patches
managing with Enterprise Manager, 8-12
physical standby database
configuration recommendations, 7-14, 7-32
restoring, 11-10
physical standby failover
using SQL*Plus, 10-11
physical standby switchover
using SQL*Plus, 10-14
primary key constraints, 7-33
production database activated
restoring, 11-18

R

RAC
benefits, 3-2
configuration recommendations, 7-12
rolling upgrade, 10-3
supported cluster system, 6-11
RAC availability notifications, 7-47
RAC instances
registering with remote listeners, 7-13
RAC only architecture, 4-2
benefits, 4-5
RAC recovery, 10-2, 10-17
unscheduled outages, 10-15
RAC rolling upgrade, 10-45
RAC rolling upgrades
recomendations, 10-48
raw device, 6-5
Real Application Clusters
benefits, 3-2
real time apply, 7-20
recommendations
component characteristics, 6-2
configuring storage, 6-2
database configuration, 7-2
redundant hardware components, 6-3
recovery catalog, 7-39
Recovery Manager
description, 3-10
using, 7-37
recovery point objective, 2-5
recovery time objective, 2-4
redo data
secure transmission, 7-31
redo log files and groups
size, 7-3
redundant hardware, 6-10
RELY constraint, 7-33
remote archiving, 7-17
remote listeners
registering instances, 7-13
REMOTE_ARCHIVE_ENABLE initialization parameter, 7-16
resetlogs on production database
restoring standby database, 11-18
restoring failed instances
RAC, 11-2
restoring failed nodes
RAC, 11-2
restoring RAC instance
client connections, 11-5
restoring service, 11-4
resumable space allocation, 7-9
RESUMABLE_TIMEOUT initialization parameter, 7-10
RETENTION GUARANTEE clause, 7-8
RETRIES parameter, 7-49
RMAN
description, 3-10
using, 7-37
RMAN autobackup, 7-39
RMAN datafile media recovery, 10-33
RMAN recovery catalog, 7-39
RMAN repository
control file, 7-42
role-based destinations, 7-17
rolling back a patch, 10-47
rolling patch upgrades, 10-45
rolling upgrade, 10-3
row and transaction inconsistencies, 10-37

RPO, 2-5
RTO, 2-4

S

SAME methodology, 6-3
scheduled outage planning, 5-9
scheduled outages
  preparation, 9-12
  primary site recovery steps, 9-9
  RAC recovery, 10-17
  secondary site recovery steps, 9-11
  types, 9-8
secondary site outage
  restoring, 11-14
secure transmission of redo data, 7-31
security policies, 5-13
security recommendations, 7-11
server hardware recommendations
  all architectures, 6-9
  Data Guard only and MAA, 6-11
  RAC only and MAA, 6-10
server parameter file, 7-12
  recovering, 10-32
server parameter file samples, N-1
server software
  recommendations for all architectures, 6-12
service level agreement
  components, 5-3
service level agreements, 2-5
service level management, 5-2
SERVICE_NAME parameter, 7-49
services
  creation, 7-50
  publishing production, 7-51
  publishing standby, 7-51
single node failure, 10-16
site failover, 10-3
  network routes, 10-5
  partial, 10-7
  recovery, 10-2
  WAN traffic managers, 6-18
SLA
  components, 5-3
SLAs, 2-5
software area, 6-4
SPFILE, 7-12
  recovering, 10-32
SPFILE samples, N-1
SQL Apply
  skipping objects, 7-35
sqlnet.ora file sample, N-8
SQLNET.SEND_BUF_SIZE and SQLNET.RECV_BUF_SIZE Oracle Net parameters, 7-22
SRLs, 7-19
SSH port forwarding, 7-30
standby archive destination, 7-17
standby control file
  recovering loss, 10-30
standby database
  comparing logical and physical, 7-14
  restoring, 11-10
standby database unique name, 7-32
standby instances
  multiple, 7-35
standby redo log
  recovering, 10-31
standby redo logs, 7-19
STANDBY_ARCHIVE_DEST initialization parameter, 7-17
storage area
  flash recovery area, 6-5
storage areas
  database area, 6-4
  software area, 6-4
storage arrays
  online servicing, 6-3
storage devices
  data validation, A-1
storage recommendations
  Data Guard only and MAA, 6-5
  RAC only and MAA, 6-5, 6-9
Streams
  description, 3-4
Streams architecture, 4-2
  benefits, 4-10
supplemental logging, 7-33
swap partitions
  mirroring, 6-12
switchover steps, 10-12

SYNC attribute
PARALLEL / NOPARALLEL, 7-27

T

table inconsistencies, 10-41
tablespace renaming, 10-54
TAF, 7-47
TCP parameters, 6-17
temporary file systems, 6-13
temporary tablespaces, 7-9
TIMED_STATISTICS initialization parameter, 7-7
tnsnames.ora file, 7-21
tnsnames.ora file sample, N-9
TRANSACTION_CONSISTENCY initialization parameter, 7-34
Transparent Application Failover, 7-47
transportable tablespace feature, 3-5

U

UNDO_MANAGEMENT initialization parameter, 7-7
UNDO_RETENTION initialization parameter, 7-7
UNDO_TABLESPACE initialization parameter, 7-7
unscheduled outages
Enterprise Manager, 8-17
primary site recovery steps, 9-4
RAC recovery, 10-15
secondary site recovery steps, 9-6
types, 9-2
upgrade
rolling for patches, 10-45
using logical standby database, 10-3, 10-49
user error
flashback technology, 10-35
recovery, 10-3

V

V$MTTR_TARGET_ADVICE view, 7-6
VALID_FOR attribute, 7-17
versions
operating system, patch, driver, 6-12

W

WAN environment
tuning the network, 7-22
WAN traffic managers, 6-18