

Oracle Database 11g: Administration Workshop II

Volume I • Student Guide

D50079GC20

Edition 2.0

September 2009

D62543

ORACLE®

Author

Maria Billings

**Technical Contributors
and Reviewers**

Christian Bauwens

Yanti Chang

Timothy Chien

Joe Fong

Andy Fortunak

Gerlinde Frenzen

Mark Fuller

Peter Fusek

Joel Goodman

Vimala Jacob

Dominique Jeunot

Pete Jones

Fukue Kawabe

Donna Keesling

Sean Kim

Achiel Langers

Gwen Lazenby

Jerry Lee

Deidre Matishak

Bill Millar

Lakshmi Naraparreddi

Ira Singer

Ranbir Singh

James Spiller

Matt Taylor

Branislav Valny

Jean-Francois Verrier

Editors

Nita Pavitran

Raj Kumar

Graphic Designer

Satish Bettegowda

Publisher

Jayanthi Keshavamurthy

Copyright © 2009, Oracle. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

I Introduction

- Course Objectives 1-2
- Suggested Schedule 1-3
- Oracle Database 11g: “g” Stands for Grid 1-4
- Grid Infrastructure for Single-Instance 1-6
- Course Examples: HR Sample Schema 1-8

1 Core Concepts and Tools of the Oracle Database

- Objectives 1-2
- Naming the Core Components of an Oracle Database Server 1-3
- Oracle Database Server Architecture Overview 1-4
- Instance-Database Configurations 1-6
- Naming the Memory Structures of an Oracle Database 1-7
- Oracle Database Memory Structures 1-8
- Process Architecture 1-10
- Process Structures 1-11
- Adding Process Names 1-13
- Process Startup Sequence 1-14
- Database Storage Architecture 1-15
- Logical and Physical Database Structures 1-17
- Automatic Storage Management 1-19
- ASM Storage Components 1-20
- ASM Instance 1-21
- DBA Configuration Tools 1-23
- Management Framework and Related DBA Tools 1-25
- Facilitating Database Management with Oracle Restart 1-26
- Notes: Facilitating Database Management with Oracle Restart 1-27
- Quiz 1-28
- Summary 1-29

2 Configuring for Recoverability

- Objectives 2-2
- Purpose of Backup and Recovery Functionality 2-3
- Typical Backup and Recovery Tasks 2-4
- Oracle Backup and Recovery Solutions 2-5

Oracle Backup Solutions	2-6
Terminology Review	2-7
What You Already Know: Oracle-Suggested Backup	2-9
Using Recovery Manager	2-10
Types of RMAN Commands	2-11
Job Commands: Example	2-12
Configuring Your Database for Backup and Recovery Operations	2-13
ARCHIVELOG Mode	2-14
Configuring ARCHIVELOG Mode	2-15
Configuring Archive Log Destinations	2-16
Guaranteeing Archive Log Success	2-17
Specifying a Retention Policy	2-19
A Recovery Window Retention Policy: Example	2-21
Using a Fast Recovery Area	2-22
Defining a Fast Recovery Area	2-24
Fast Recovery Area Space Management	2-25
Fast Recovery Area Space Usage	2-27
What Is Done Automatically for You	2-29
Monitoring the FRA	2-30
Benefits of Using a Fast Recovery Area	2-31
Quiz	2-32
Summary	2-34
Practice 2 Overview: Configuring for Recoverability	2-35

3 Using the RMAN Recovery Catalog

Objectives	3-2
RMAN Repository Data Storage: Comparison of Options	3-3
Storing Information in the Recovery Catalog	3-4
Reasons to Use a Recovery Catalog	3-5
Creating the Recovery Catalog: Three Steps	3-6
Configuring the Recovery Catalog Database	3-7
Creating the Recovery Catalog Owner	3-8
Creating the Recovery Catalog	3-9
Managing Target Database Records in the Recovery Catalog	3-10
Registering a Database in the Recovery Catalog	3-11
Using Enterprise Manager to Register a Database	3-12
Unregistering a Target Database from the Recovery Catalog	3-13
Cataloging Additional Backup Files	3-14
Recovery Catalog Resynchronization: Concepts	3-16
Manually Resynchronizing the Recovery Catalog	3-17
Using RMAN Stored Scripts	3-18

- Executing RMAN Stored Scripts 3-19
- Maintaining RMAN Stored Scripts 3-20
- Backing Up the Recovery Catalog 3-21
- Re-Creating an Unrecoverable Recovery Catalog 3-22
- Exporting and Importing the Recovery Catalog 3-23
- Upgrading and Dropping the Recovery Catalog 3-24
- IMPORT CATALOG Command 3-25
- Creating and Using Virtual Private Catalogs 3-27
- Using RMAN Virtual Private Catalogs 3-28
- Recovery Catalogs Summary 3-30
- Quiz 3-32
- Summary 3-34
- Practice 3 Overview: Using the RMAN Recovery Catalog 3-35
- 4 Configuring Backup Settings**
 - Objectives 4-2
 - Configuring Persistent Settings for RMAN 4-3
 - Viewing Persistent Settings 4-4
 - Control File Autobackups 4-5
 - Managing Persistent Settings 4-7
 - Using a Media Manager 4-8
 - Specifying a Backup Destination 4-10
 - Configuring and Allocating Channels 4-11
 - Creating Duplexed Backup Sets 4-12
 - Creating Duplexed Backup Sets Using CONFIGURE BACKUP COPIES 4-13
 - Backup Optimization 4-14
 - Saving Backup Space with Unused Block Compression 4-16
 - Compressing Backups 4-17
 - Using RMAN Backup Compression 4-18
 - Encrypting Backups 4-19
 - Quiz 4-20
 - Summary 4-22
 - Practice 4 Overview: Configuring Backup Specifications 4-23
- 5 Creating Backups with RMAN**
 - Objectives 5-2
 - Creating Backup Sets 5-3
 - Creating Image Copies 5-4
 - Creating a Whole Database Backup 5-6
 - RMAN Backup Types 5-8
 - Fast Incremental Backup 5-10

Enabling Fast Incremental Backup	5-11
Monitoring Block Change Tracking	5-12
Performing Proxy Copies	5-13
Creating Duplexed Backup Sets Using <code>BACKUP COPIES</code>	5-14
Creating Backups of Backup Sets	5-15
Backing Up Read-Only Tablespaces	5-16
Configuring Backup and Restore for Very Large Files	5-17
Creating RMAN Multisection Backups	5-18
Archival Backups: Concepts	5-19
Creating Archival Backups with EM	5-21
Creating Archival Backups with RMAN	5-22
Managing Archival Database Backups	5-23
Backing Up Recovery Files	5-24
Managing Backups: Reporting	5-25
Managing Backups: Dynamic Performance Views	5-27
Using Enterprise Manager to View Backup Reports	5-28
Managing Backups: Cross-Checking and Deleting	5-29
Quiz	5-30
Summary	5-32
Practice 5 Overview: Creating Backups	5-33

6 Restore and Recovery Tasks

Objectives	6-2
Restoring and Recovering	6-3
Causes of File Loss	6-4
Critical Versus Noncritical	6-5
Automatic Tempfile Recovery	6-6
Log Group Status: Review	6-7
Recovering from the Loss of a Redo Log Group	6-8
Clearing a Log File	6-9
Recovering from a Lost Index Tablespace	6-10
Re-Creating Indexes	6-11
Authentication Methods for Database Administrators	6-13
Re-creating a Password Authentication File	6-14
Comparing Complete and Incomplete Recovery	6-16
Complete Recovery Process	6-17
Point-in-Time Recovery Process	6-18
Recovering a Read-Only Tablespace	6-20
Recovering <code>NOLOGGING</code> Database Objects	6-21
Recovering from the Loss of All Control File Copies: Overview	6-22
Recovering the Control File to the Default Location	6-23

Quiz 6-24

Summary 6-26

7 Using RMAN to Perform Recovery

Objectives 7-2

Using RMAN RESTORE and RECOVER Commands 7-3

Performing Complete Recovery: Loss of a Noncritical Data File in ARCHIVELOG Mode 7-4

Performing Complete Recovery: Loss of a System-Critical Data File in ARCHIVELOG Mode 7-5

Recovering Image Copies 7-6

Recovering Image Copies: Example 7-7

Performing a Fast Switch to Image Copies 7-8

Using SET NEWNAME for Switching Files 7-9

Substitution Variables for SET NEWNAME 7-10

Performing Restore and Recovery of a Database in NOARCHIVELOG Mode 7-11

Using Restore Points 7-12

Performing Point-in-Time Recovery 7-13

Performing Recovery with a Backup Control File 7-15

Recovery from Loss of Server Parameter File 7-16

Restoring the Server Parameter File from the Control File Autobackup 7-17

Restoring the Control File from Autobackup 7-18

Using Incremental Backups to Recover a Database in NOARCHIVELOG Mode 7-20

Restoring and Recovering the Database on a New Host 7-21

Preparing to Restore the Database to a New Host 7-22

Restoring the Database to a New Host 7-23

Performing Disaster Recovery 7-27

Quiz 7-29

Summary 7-31

Practice 7 Overview: Using RMAN to Perform Recovery 7-32

8 Monitoring and Tuning RMAN

Objectives 8-2

Parallelization of Backup Sets 8-3

Monitoring RMAN Sessions 8-5

Monitoring RMAN Job Progress 8-7

Interpreting RMAN Message Output 8-9

Using the DEBUG Option 8-10

Interpreting RMAN Error Stacks 8-11

Tuning RMAN 8-12

- RMAN Multiplexing 8-14
- Allocating Disk Buffers: Example 8-15
- Allocating Tape Buffers 8-16
- Comparing Synchronous and Asynchronous I/O 8-18
- Monitoring RMAN Job Performance 8-20
- Asynchronous I/O Bottlenecks 8-21
- Synchronous I/O Bottlenecks 8-22
- Channel Tuning 8-23
- Tuning the BACKUP Command 8-25
- Tuning RMAN Backup Performance 8-27
- Setting LARGE_POOL_SIZE 8-28
- Tuning RMAN Tape Streaming Performance Bottlenecks 8-29
- Quiz 8-31
- Summary 8-33
- Practice 8 Overview: Monitoring and Tuning RMAN 8-34

9 Diagnosing the Database

- Objectives 9-2
- Data Recovery Advisor 9-3
- Data Failures 9-6
- Data Failure: Examples 9-7
- Data Recovery Advisor RMAN Command-Line Interface 9-8
- Listing Data Failures 9-9
- Advising on Repair 9-11
- Executing Repairs 9-12
- Classifying (and Closing) Failures 9-13
- Data Recovery Advisor Views 9-14
- Best Practice: Proactive Checks 9-15
- What Is Block Corruption? 9-16
- Block Corruption Symptoms: ORA-01578 9-17
- How to Handle Corruption 9-18
- Setting Parameters to Detect Corruption 9-19
- Block Media Recovery 9-21
- Prerequisites for Block Media Recovery 9-22
- The RECOVER . . . BLOCK Command 9-23
- Automatic Diagnostic Workflow 9-24
- Automatic Diagnostic Repository 9-25
- The ADR Command-Line Tool (ADRCI) 9-26
- The V\$DIAG_INFO View 9-27
- Location for Diagnostic Traces 9-28

Health Monitor: Overview	9-29
Running Health Checks Manually: PL/SQL Example	9-30
Viewing HM Reports Using the ADRCI Utility	9-31
Quiz	9-32
Summary	9-36
Practice 9 Overview: Diagnosing the Database	9-37
10 Using Flashback Technology I	
Objectives	10-2
Flashback Technology	10-3
Transactions and Undo	10-4
Guaranteeing Undo Retention	10-5
Preparing Your Database for Flashback	10-6
Using Flashback Technology to Query Data	10-8
Flashback Query	10-9
Flashback Query: Example	10-10
Flashback Version Query	10-11
Flashback Version Query: Considerations	10-12
Quiz	10-13
Flashback Table: Overview	10-15
Flashback Table	10-16
Enabling Row Movement on a Table	10-17
Performing Flashback Table	10-18
Flashback Table: Considerations	10-19
Quiz	10-20
Flashback Transaction Query	10-21
Using Enterprise Manager to Perform Flashback Transaction Query	10-22
Flashback Transaction Query: Considerations	10-23
Flashback Transaction	10-24
Prerequisites	10-25
Flashing Back a Transaction	10-26
Possible Workflow	10-27
Flashback Transaction Wizard	10-28
Choosing Other Back-out Options	10-29
Final Steps Without EM	10-31
Quiz	10-32
Summary	10-33
Practice 10 Overview: Performing Flashback Transaction Backout	10-34

11 Using Flashback Technology II

- Objectives 11-2
- Oracle Total Recall Overview 11-3
- Setup Process 11-4
- How Total Recall Works 11-5
- Oracle Total Recall Scenario 11-6
- Transparent Schema Evolution 11-8
- Full Schema Evolution 11-9
- Restrictions 11-10
- Guidelines 11-11
- Viewing Flashback Data Archives 11-12
- Quiz 11-13
- Flashback Drop and the Recycle Bin 11-15
- Recycle Bin 11-16
- Restoring Tables from the Recycle Bin 11-18
- Recycle Bin: Automatic Space Reclamation 11-19
- Recycle Bin: Manual Space Reclamation 11-20
- Bypassing the Recycle Bin 11-21
- Querying the Recycle Bin 11-22
- Quiz 11-23
- Summary 11-24
- Practice 11 Overview: Using Flashback Technology 11-25

12 Performing Flashback Database

- Objectives 12-2
- Flashback Database 12-3
- Flashback Database Architecture 12-4
- Configuring Flashback Database 12-5
- What You Need to Do 12-6
- Flashback Database: Examples 12-7
- Flashback Database Considerations 12-8
- Monitoring Flashback Database 12-9
- Monitoring Flashback Database with EM 12-11
- Guaranteed Restore Points 12-12
- Flashback Database and Guaranteed Restore Points 12-13
- Quiz 12-15
- Summary 12-17
- Practice 12 Overview: Working with Flashback Database 12-18

13 Managing Memory

- Objectives 13-2
- Memory Management: Overview 13-3
- Reviewing Oracle Database Memory Structures 13-4
- Buffer Cache 13-6
- Using Multiple Buffer Pools 13-8
- Shared Pool 13-10
- Large Pool 13-11
- Java Pool and Streams Pool 13-12
- Redo Log Buffer 13-13
- Automatic Memory Management: Overview 13-14
- Oracle Database Memory Parameters 13-15
- Monitoring Automatic Memory Management 13-16
- Efficient Memory Usage: Guidelines 13-18
- Memory Tuning Guidelines for the Library Cache 13-20
- Automatic Shared Memory Management: Overview 13-22
- How ASMM Works 13-23
- Enabling Automatic Shared Memory Management 13-24
- Disabling ASMM 13-25
- Program Global Area (PGA) 13-26
- Using the `V$PARAMETER` View 13-28
- Quiz 13-29
- Summary 13-30
- Practice 13 Overview: Using AMM to Correct a Memory Allocation Problem 13-31

14 Managing Database Performance

- Objectives 14-2
- Tuning Activities 14-3
- Performance Planning 14-4
- Instance Tuning 14-6
- Performance Tuning Methodology 14-7
- Performance Monitoring 14-8
- Performance Tuning Data 14-9
- Optimizer Statistics Collection 14-10
- Statistic Preferences: Overview 14-12
- Using Statistic Preferences 14-13
- Setting Global Preferences with Enterprise Manager 14-14
- Oracle Wait Events 14-15
- Instance Statistics 14-16
- Monitoring Session Performance 14-18
- Displaying Session-Related Statistics 14-19

Displaying Service-Related Statistics	14-20
Troubleshooting and Tuning Views	14-21
Dictionary Views	14-22
Automatic Workload Repository	14-23
Using Automatic Workload Repository Views	14-25
Real Application Testing Overview: Database Replay	14-26
The Big Picture	14-27
Quiz	14-28
Summary	14-29
Practice 14 Overview: Monitoring Instance Performance	14-30

15 Managing Performance by SQL Tuning

Objectives	15-2
SQL Tuning	15-3
SQL Advisors	15-4
Automatic SQL Tuning Results	15-5
Implement Automatic Tuning Recommendations	15-6
SQL Tuning Advisor: Overview	15-7
Using the SQL Tuning Advisor	15-8
SQL Tuning Advisor Options	15-9
SQL Tuning Advisor Recommendations	15-10
Using the SQL Tuning Advisor: Example	15-11
Duplicate SQL	15-12
SQL Access Advisor: Overview	15-13
Typical SQL Access Advisor Session	15-14
Workload Source	15-15
Recommendation Options	15-16
Reviewing Recommendations	15-18
SQL Performance Analyzer: Overview	15-19
SQL Performance Analyzer: Use Cases	15-20
Using SQL Performance Analyzer	15-21
Quiz	15-22
Summary	15-26
Practice 15 Overview: Managing Performance by SQL Tuning	15-27

16 Managing Resources

Objectives	16-2
Database Resource Manager: Overview	16-3
Database Resource Manager: Concepts	16-4
Why Use Resource Manager	16-5
Default Maintenance Resource Manager Plan	16-7

Example: <code>DEFAULT_PLAN</code>	16-8
Potential Work Flow	16-9
Specifying Resource Plan Directives	16-11
Resource Allocation Methods for Resource Plans	16-12
Comparison of <code>EMPHASIS</code> and <code>RATIO</code>	16-13
Active Session Pool Mechanism	16-15
Setting the Active Session Pool	16-16
Specifying Thresholds	16-18
Setting Idle Timeouts	16-19
Limiting CPU Utilization at the Database Level	16-20
Limiting CPU Utilization at the Server Level: Instance Caging	16-22
Instance Caging Examples	16-23
Monitoring Instance Caging	16-24
Resource Consumer Group Mapping	16-25
Activating a Resource Plan	16-27
Database Resource Manager Information	16-28
Monitoring the Resource Manager	16-29
Quiz	16-32
Summary	16-33
Practice 16 Overview: Using the Resource Manager	16-34
17 Automating Tasks with the Scheduler	
Objectives	17-2
Simplifying Management Tasks	17-3
Core Components	17-4
Your Basic Work Flow	17-5
Quiz	17-7
Persistent Lightweight Jobs	17-8
Using a Time-Based or Event-Based Schedule	17-9
Creating a Time-Based Job	17-10
Creating an Event-Based Schedule	17-12
Creating Event-Based Schedules with Enterprise Manager	17-13
Creating an Event-Based Job	17-14
Event-Based Scheduling	17-15
Creating Complex Schedules	17-17
Quiz	17-18
Using Email Notification	17-19
Adding and Removing Email Notifications	17-20
Creating Job Chains	17-21
Example of a Chain	17-23
Advanced Scheduler Concepts	17-24

- Job Classes 17-25
- Windows 17-27
- Prioritizing Jobs Within a Window 17-28
- Creating a Job Array 17-29
- Quiz 17-31
- Creating a File Watcher and an Event-Based Job 17-32
- Enabling File Arrival Events from Remote Systems 17-34
- Scheduling Remote Database Jobs 17-35
- Creating Remote Database Jobs 17-36
- Scheduling Multiple Destination Jobs 17-37
- Viewing Scheduler Meta Data 17-38
- Quiz 17-40
- Summary 17-41
- Practice 17 Overview: Automating Tasks with the Scheduler 17-42

18 Managing Space

- Objectives 18-2
- Space Management: Overview 18-3
- Block Space Management 18-4
- Row Chaining and Migration 18-5
- Quiz 18-7
- Free Space Management Within Segments 18-8
- Types of Segments 18-9
- Allocating Extents 18-10
- Allocating Space 18-11
- Creating Tables Without Segments 18-12
- Controlling Deferred Segment Creation 18-13
- Restrictions and Exceptions 18-14
- Additional Automatic Functionality 18-15
- Quiz 18-16
- Table Compression: Overview 18-17
- Compression for Direct-Path Insert Operations 18-18
- OLTP Compression for DML Operations 18-20
- Specifying Table Compression 18-21
- Using the Compression Advisor 18-22
- Using the DBMS_COMPRESSION Package 18-23
- Compressing Table Data 18-24
- Proactive Tablespace Monitoring 18-25
- Thresholds and Resolving Space Problems 18-26
- Monitoring Tablespace Space Usage 18-27
- Shrinking Segments 18-28

Results of Shrink Operation	18-29
Reclaiming Space Within ASSM Segments	18-30
Segment Advisor: Overview	18-31
Segment Advisor	18-32
Implementing Recommendations	18-33
Automatic Segment Advisor	18-34
Manual Segment Shrink Using EM	18-35
Shrinking Segments Using SQL	18-36
Managing Resumable Space Allocation	18-37
Using Resumable Space Allocation	18-38
Resuming Suspended Statements	18-40
What Operations Are Resumable?	18-42
Quiz	18-43
Summary	18-44
Practice 18 Overview: Managing Storage	18-45
19 Managing Space for the Database	
Objectives	19-2
Database Storage	19-3
Supporting 4-KB Sector Disks	19-4
Using 4-KB Sector Disks	19-5
Specifying the Disk Sector Size	19-6
Quiz	19-7
Transporting Tablespaces	19-10
Concept: Minimum Compatibility Level	19-11
Minimum Compatibility Level	19-12
Transportable Tablespace Procedure	19-13
Determining the Endian Format of a Platform	19-14
Using the RMAN CONVERT Command	19-16
Transportable Tablespaces with Enterprise Manager	19-17
Transporting Databases	19-20
Database Transportation Procedure: Source System Conversion	19-21
Database Transportation Procedure: Target System Conversion	19-22
Database Transportation: Considerations	19-23
Quiz	19-24
Summary	19-25
Practice 19 Overview: Managing Space for the Database	19-26

20 Duplicating a Database

- Objectives 20-2
- Using a Duplicate Database 20-3
- Choosing Database Duplication Techniques 20-4
- Duplicating an Active Database 20-5
- Duplicating a Database with a Target Connection 20-6
- Duplicating a Database with Recovery Catalog Without Target Connection 20-7
- Duplicating a Database Without Recovery Catalog or Target Connection 20-8
- Creating a Backup-Based Duplicate Database 20-9
- Creating an Initialization Parameter File for the Auxiliary Instance 20-10
- Specifying New Names for Your Destination 20-11
- Using the `SET NEWNAME` Clauses 20-12
- Substitution Variables for `SET NEWNAME` 20-13
- Specifying Parameters for File Naming 20-14
- Starting the Instance in `NOMOUNT` Mode 20-16
- Ensuring That Backups and Archived Redo Log Files Are Available 20-17
- Allocating Auxiliary Channels 20-18
- Understanding the RMAN Duplication Operation 20-19
- Specifying Options for the `DUPLICATE` Command 20-21
- Using Additional `DUPLICATE` Command Options 20-22
- Using EM to Clone a Database 20-23
- Quiz 20-24
- Summary 20-25
- Practice 20 Overview: Duplicating a Database 20-26

Appendix A: Practices and Solutions

Appendix B: Performing Tablespace Point-in-Time Recovery

- Objectives B-2
- Tablespace Point-in-Time Recovery (TSPITR): Concepts B-3
- Tablespace Point-in-Time Recovery (TSPITR): Terminology B-4
- Tablespace Point-in-Time Recovery: Architecture B-5
- When to Use TSPITR B-7
- Preparing for TSPITR B-8
- Determining the Correct Target Time B-9
- Determining the Tablespaces for the Recovery Set B-10
- Identifying Relationships That Span Recovery Set Boundaries B-11
- Identifying Objects That Will Be Lost B-12
- Performing Basic RMAN TSPITR B-13
- Performing Fully Automated TSPITR B-14

Using Image Copies for Faster TSPITR Performance	B-15
Using Enterprise Manager to Perform TSPITR	B-16
RMAN TSPITR Processing	B-17
Performing RMAN TSPITR with an RMAN-Managed Auxiliary Instance	B-19
Performing RMAN TSPITR Using Your Own Auxiliary Instance	B-20
Troubleshooting RMAN TSPITR	B-21
Summary	B-22

Appendix C: Performing User-Managed Backup and Recovery

Objectives	C-2
Types of Backup and Recovery Practices	C-3
Performing a User-Managed Backup of the Database	C-4
The Need for Backup Mode	C-5
Identifying Files to Manually Backup	C-6
Manually Backing Up a NOARCHIVELOG Database	C-7
Manually Backing Up an ARCHIVELOG Database	C-8
Backing Up the Control File	C-9
Performing User-Managed Complete Database Recovery: Overview	C-10
Performing Complete Closed Database Recovery: Overview	C-11
Identifying Recovery-Related Files	C-12
Restoring Recovery-Related Files	C-13
Applying Redo Data	C-15
Performing Complete Open Database Recovery	C-16
Performing User-Managed Incomplete Recovery: Overview	C-18
Choosing an Incomplete Recovery Method	C-19
Performing User-Managed Incomplete Recovery	C-20
Performing User-Managed Incomplete Recovery: Steps	C-22
User-Managed Time-Based Recovery: Example	C-23
User-Managed Cancel-Based Recovery: Example	C-25
Summary	C-27

Appendix D: Managing the ASM Instance

Objectives	D-2
ASM Benefits for Administrators	D-3
ASM Instance	D-4
ASM Components: ASM Instance—Primary Processes	D-6
ASM Instance Initialization Parameters	D-7
Interaction Between Database Instances and ASM	D-9
ASM Instance: Dynamic Performance Views	D-10
ASM System Privileges	D-11
Using Enterprise Manager to Manage ASM Users	D-12

Starting and Stopping ASM Instances Using SQL*Plus	D-13
Starting and Stopping ASM Instances Using <code>srvctl</code>	D-15
Starting and Stopping ASM Instances Using <code>asmcmd</code>	D-16
Disk Group Overview	D-17
ASM Disks	D-18
Allocation Units	D-19
ASM Files	D-20
Extent Maps	D-21
Striping Granularity	D-22
Fine Grained Striping	D-23
ASM Failure Groups	D-25
Stripe and Mirror Example	D-26
Failure Example	D-27
Managing Disk Groups	D-28
Creating and Dropping Disk Groups Using SQL*Plus	D-29
Adding Disks to Disk Groups	D-30
Miscellaneous <code>ALTER</code> Commands	D-31
ASM Management Using Enterprise Manager	D-32
ASM Disk Group Compatibility	D-33
ASM Disk Group Attributes	D-35
Using Enterprise Manager to Edit Disk Group Attributes	D-36
Retrieving ASM Metadata	D-37
ASM Fast Mirror Resync Overview	D-38
Summary	D-39

I Introduction

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Course Objectives

After completing this course, you should be able to:

- Configure the Oracle Database for optimal recovery
- Back up and recover a database (and its parts) with Recovery Manager (RMAN)
- Use flashback technology to view past states of data and to revert objects to a past state
- Identify burdensome database sessions and poorly performing SQL
- Use an appropriate and flexible memory configuration
- Configure resource allocations among sessions and tasks
- Schedule jobs to run inside or outside the database
- Use compression to optimize database storage and duplicate a database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Course Objectives

In this course, you learn to:

- Secure the availability of your database by using appropriate backup and recovery strategies
- Diagnose and repair data failures with flashback technology
- Monitor and manage major database components, including memory, performance, and resources
- Automate DBA tasks with the scheduler
- Manage space to optimize database storage and to be able to respond to growing space requirements

Suggested Schedule

Day	Lessons	Day	Lessons
1	<ol style="list-style-type: none"> 1. Core Concepts and Tools of the Oracle Database 2. Configuring for Recoverability 3. Using the RMAN Recovery Catalog 4. Configuring Backup Settings 	4	<ol style="list-style-type: none"> 13. Managing Memory 14. Managing Database Performance 15. Managing Performance by SQL Tuning 16. Managing Resources 17. Automating Tasks with the Scheduler
2	<ol style="list-style-type: none"> 5. Creating Backups with RMAN 6. Restore and Recovery Tasks 7. Using RMAN to Perform Recovery 8. Monitoring and Tuning RMAN 	5	<ol style="list-style-type: none"> 18. Managing Space 19. Managing Space for the Database 20. Duplicating a Database
3	<ol style="list-style-type: none"> 9. Diagnosing the Database 10. Using Flashback Technology I 11. Using Flashback technology II 12. Performing Flashback Database 		

ORACLE

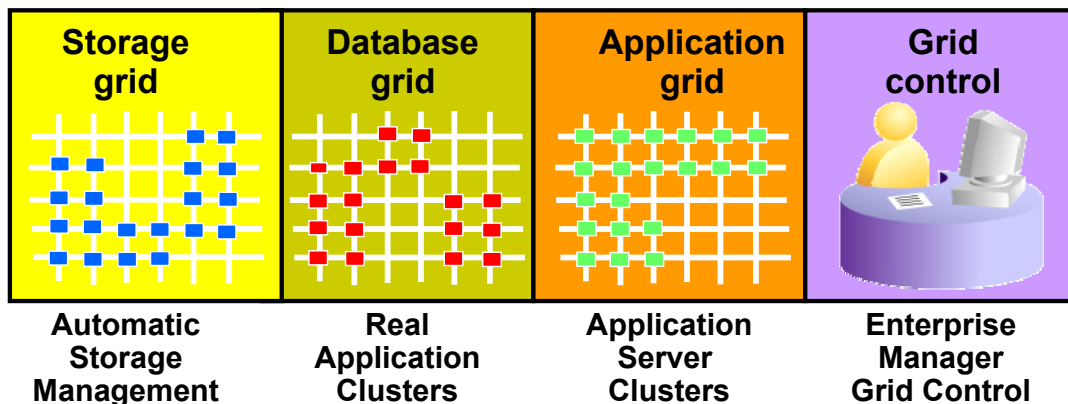
Copyright © 2009, Oracle. All rights reserved.

Suggested Schedule

This schedule is just a very general outline. Your instructor will determine the actual class schedule.

Oracle Database 11g: “g” Stands for Grid

- Open Grid Forum (OGF)
- Oracle’s grid infrastructure:
 - Low cost
 - High quality of service
 - Easy to manage



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Database 11g: “g” Stands for Grid

Open Grid Forum (OGF) is a standards body that develops standards for grid computing. It comprises a set of committees and working groups that focus on various aspects of grid computing. The committees and working groups are composed of participants from academia, the research community, and (increasingly) commercial companies. You can see the website of OGF at <http://www.ogf.org>.

Oracle has created the grid computing infrastructure software that balances all types of workloads across servers and enables all those servers to be managed as one complete system. Grid computing can achieve the same very high level of reliability as mainframe computing because all components are clustered. But unlike mainframes and large UNIX symmetric multiprocessing (SMP) servers, a grid can be built with open system technologies, such as Intel processors and the Linux operating system, at a very low cost.

Oracle’s grid computing technology includes:

- Automatic Storage Management (ASM)
- Real Application Clusters (RAC)
- Application Server Clusters
- Enterprise Manager Grid Control

Oracle Database 11g: “g” Stands for Grid (continued)

Automatic Storage Management spreads database data across all disks, creates and maintains a storage grid, and provides the highest input/output (I/O) throughput with minimal management costs. As disks are added or dropped, ASM redistributes the data automatically. (There is no need for a logical volume manager to manage the file system.) Data availability increases with optional mirroring, and you can add or drop disks online. See the lesson titled “Managing Database Storage Structures.”

Oracle’s **Real Application Clusters** runs and scales all application workloads on a cluster of servers and offers the following features:

- **Integrated clusterware:** This includes functionality for cluster connectivity, messaging and locking, cluster control, and recovery. It is available on all platforms that are supported by Oracle Database 10g.
- **Automatic workload management:** Rules can be defined to automatically allocate processing resources to each service both during normal operations and in response to failures. These rules can be dynamically modified to meet the changing business needs. This dynamic resource allocation within a database grid is unique to Oracle RAC.
- **Automatic event notification to the mid-tier:** When a cluster configuration changes, the mid-tier can immediately adapt to instance failover or availability of a new instance. This enables end users to continue working in the event of instance failover without the delays typically caused by network timeouts. In the event of new instance availability, the mid-tier can immediately start load balancing connections to that instance. Oracle Database 10g Java Database Connectivity (JDBC) drivers have the “fast connection failover” functionality that can be automatically enabled to handle these events.

Oracle WebLogic Application Grid works with any application server - including Oracle WebLogic Server, IBM WebSphere Application Server, and JBoss Application Server - or in a pure grid environment without an application server. Oracle WebLogic Application Grid provides extreme and predictable application scalability and performance. With capacity on demand, Oracle WebLogic Application Grid can linearly scale out middleware infrastructure from a few to thousands of servers. Through its in-memory data grid solution, it provides fast access to frequently used data. Leveraging this grid capability, computation can be done in parallel, further improving application performance.

Enterprise Manager Grid Control manages gridwide operations that include managing the entire stack of software, provisioning users, cloning databases, and managing patches. It can monitor the performance of all applications from the point of view of your end users. Grid Control views the performance and availability of the grid infrastructure as a unified whole rather than as isolated storage units, databases, and application servers. You can group hardware nodes, databases, and application servers into single logical entities and manage a group of targets as one unit.

Note: In this course, you use Enterprise Manager Database Console to manage one database at a time.

Grid Infrastructure for Single-Instance

Grid Infrastructure for Single-Instance is introduced with Oracle Database 11g Release 2 (11.2)

- Is installed from the clusterware media, separate from the Oracle database software
- Contains Oracle Automatic Storage Management (ASM)
- Contains Oracle Restart—a high availability solution for nonclustered databases
 - Can monitor and restart the following components:
 - Database instances
 - Oracle Net listener
 - Database services
 - ASM instance
 - ASM disk groups
 - Oracle Notification Services (ONS/eONS) for Data Guard

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Grid infrastructure for Single-Instance

Grid Infrastructure for Single-Instance is introduced with Oracle Database 11g Release 2. It is installed from the clusterware media, separate from the Oracle database software and now includes Oracle Automatic Storage Management and a new feature called Oracle Restart.

Oracle Restart is designed to improve the availability of your Oracle Database. It implements a high availability solution for single instance (nonclustered) environments only. For Oracle Real Application Cluster (Oracle RAC) environments, the functionality to automatically restart components is provided by Oracle Clusterware. Oracle Restart can monitor the health and automatically restart the following components:

- Database Instances
- Oracle Net Listener
- Database Services
- ASM Instance
- ASM Disk Groups
- Oracle Notification Services (ONS/eONS) for Data Guard

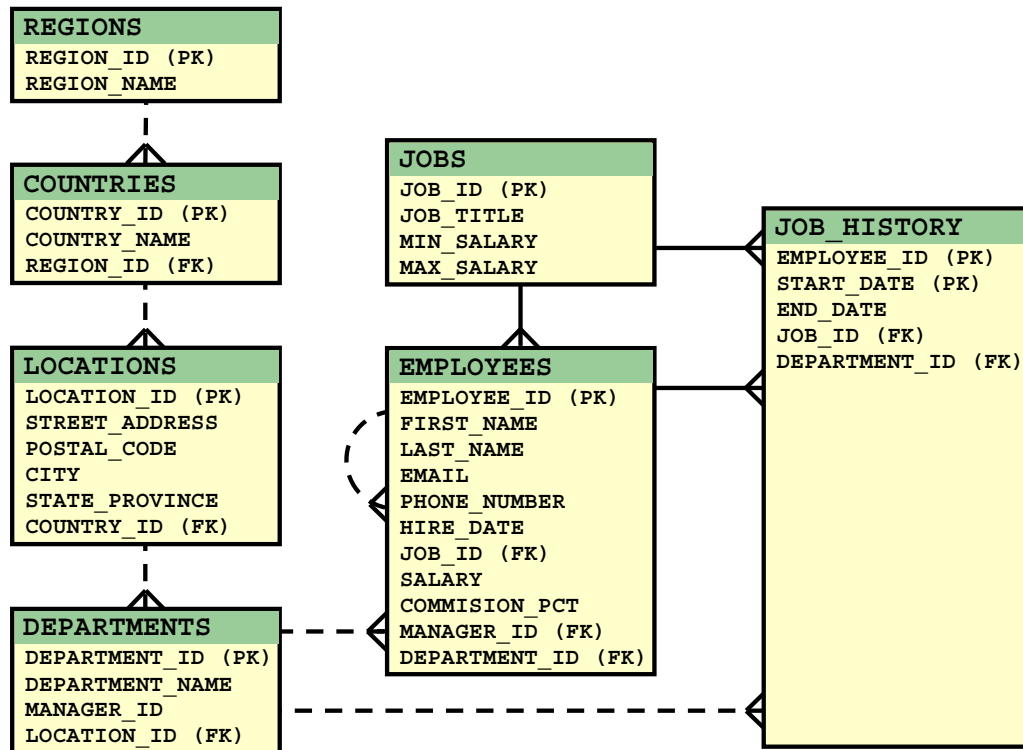
Oracle Restart ensures that the components are started in the proper order, in accordance with component dependencies. If a component must be shut down, it ensures that the dependent components are cleanly shut down first. Oracle Restart runs out of the Oracle Grid Infrastructure home, which you install separately from Oracle Database homes.

Grid infrastructure for Single-Instance (continued)

Some glossary definitions (for your ease of reference):

- A **database instance** is the combination of the system global area (SGA) and background processes. An instance is associated with one and only one database. In an Oracle Real Application Clusters configuration, multiple instances access a single database simultaneously.
- An **Oracle Net listener** is a process that listens for incoming client connection requests and manages network traffic to the database.
- A **database service** is a user-created service that is managed by Oracle Clusterware. A database service may be offered on one or more RAC instances, and managed on per-instance basis (with respect to starting/stopping the service). Only services that are managed by Oracle Clusterware are able to be part of a Performance Class. Services created with the DBMS_SERVICE package are not managed by Oracle Clusterware.
- An **ASM instance** is built on the same technology as an Oracle Database instance. An ASM instance has a System Global Area (SGA) and background processes that are similar to those of Oracle Database. However, because ASM performs fewer tasks than a database, an ASM SGA is much smaller than a database SGA. ASM instances mount disk groups to make ASM files available to database instances; ASM instances do not mount databases.
- An **ASM disk groups** consists of one or more ASM disks, which are managed as a logical unit. I/O to a disk group is automatically spread across all the disks in the group.
- An **Oracle Notification Services (ONS)** is a publish-and-subscribe service for communicating information about all Fast Application Notification (FAN) events.

Course Examples: HR Sample Schema



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Course Examples: HR Sample Schema

The examples used in this course are from a human resources (HR) application, which can be created as part of the starter database.

The following are some principal business rules of the HR application:

- Each department may be the employer of one or more employees. Each employee may be assigned to only one department.
- Each job must be a job for one or more employees. Each employee must be currently assigned to only one job.
- When an employee changes his or her department or job, a record in the JOB_HISTORY table records the start and end dates of the past assignments.
- JOB_HISTORY records are identified by a composite primary key (PK): the EMPLOYEE_ID and the START_DATE columns.

Notation: PK = Primary Key, FK = Foreign Key

Solid lines represent mandatory foreign key (FK) constraints and dashed lines represent optional FK constraints.

The EMPLOYEES table also has an FK constraint with itself. This is an implementation of the business rule: Each employee may be reporting directly to only one manager. The FK is optional because the top employee does not report to another employee.

1

Core Concepts and Tools of the Oracle Database

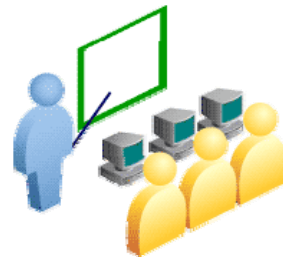
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the core concepts of the Oracle Database architecture with Automatic Storage Management (ASM)
- Use configuration and management DBA tools
- Describe the technical course environment



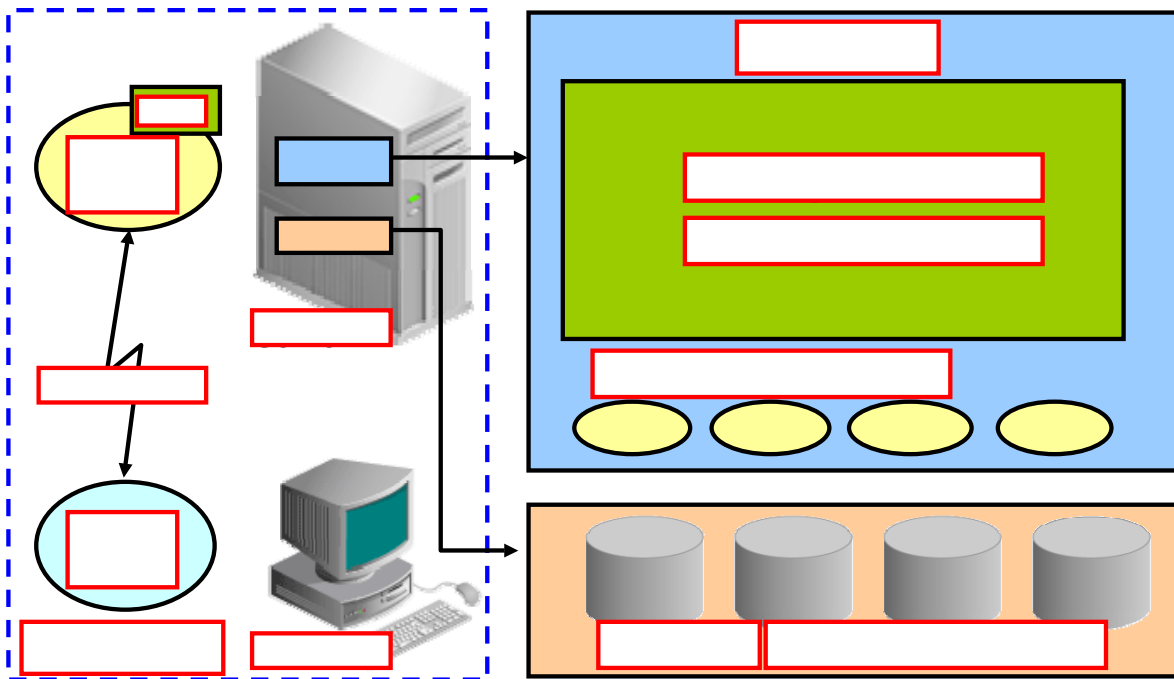
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

This lesson reviews the Oracle Database architecture with ASM and provides an overview of the technical environment for this course. In this way, core concepts and DBA tools are introduced.

Naming the Core Components of an Oracle Database Server



ORACLE

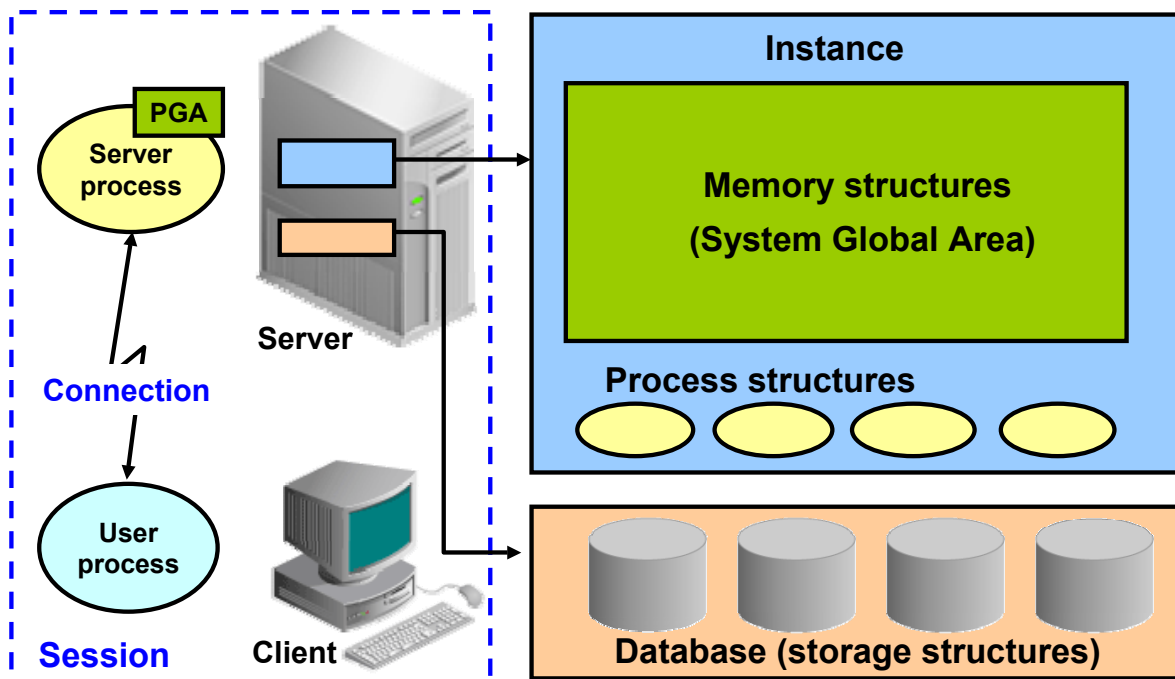
Copyright © 2009, Oracle. All rights reserved.

Naming the Core Components of an Oracle Database Server

The following are a few sample questions to get you started by naming the core components:

1. The two main components of a basic Oracle Database system: _____ and _____.
2. The Instance consists of _____ and _____ processes.
3. The three major structures in the Oracle Database server architecture are: _____, _____, and _____.
4. A session is a connection between the _____ and the _____.

Oracle Database Server Architecture Overview



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Database Server Architecture

There are three major structures in the Oracle Database server architecture: memory structures, process structures, and storage structures. A basic Oracle database system consists of an Oracle database and a database instance.

The database consists of both physical structures and logical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting access to logical storage structures.

The instance consists of memory structures and background processes associated with that instance. Every time an instance is started, a shared memory area called the System Global Area (SGA) is allocated and the background processes are started. Processes are jobs that work in the memory of computers. A process is defined as a “thread of control” or a mechanism in an operating system that can run a series of steps. After starting a database instance, the Oracle software associates the instance with a specific database. This is called *mounting the database*. The database is then ready to be opened, which makes it accessible to authorized users.

Note: The Oracle Automatic Storage Management (ASM) uses the concept of an instance for the memory and process components, but is not associated with a specific database.

Connections and sessions are closely related to user processes but are very different in meaning.

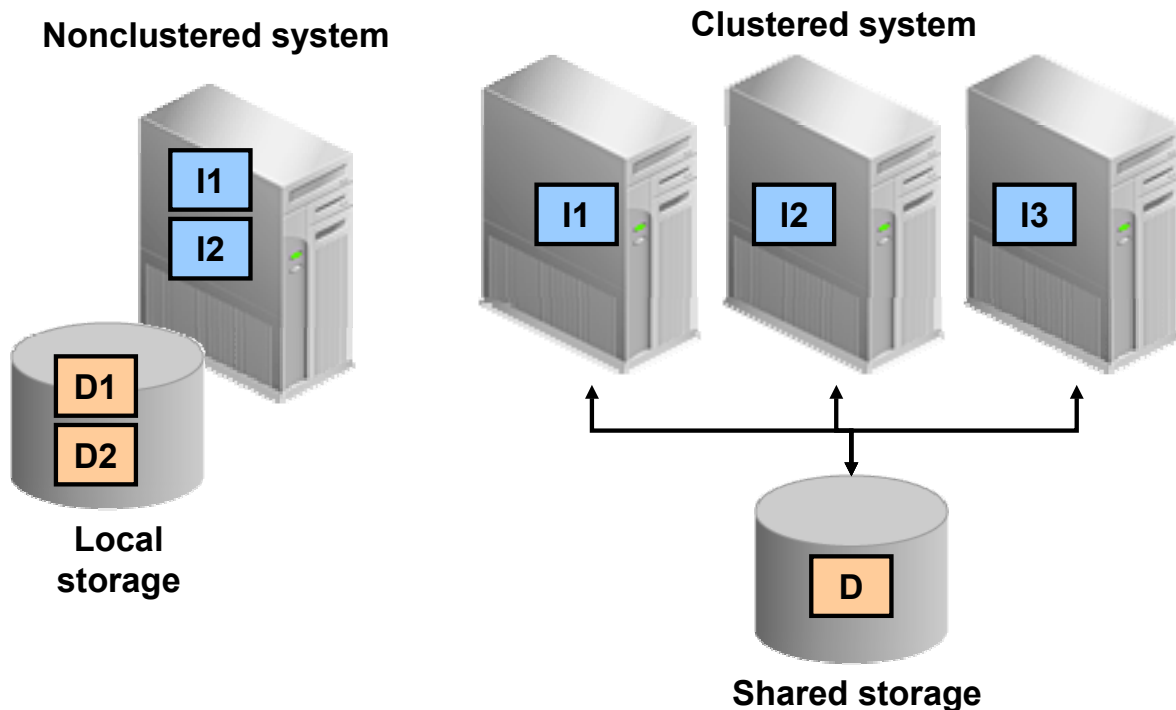
Oracle Database Architecture Overview (continued)

A *connection* is a communication pathway between a user process and an Oracle Database instance. A communication pathway is established using available interprocess communication mechanisms (on a computer that runs both the user process and Oracle Database) or network software (when different computers run the database application and Oracle Database, and communicate through a network).

A *session* represents the state of a current user login to the database instance. For example, when a user starts SQL*Plus, the user must provide a valid username and password, and then a session is established for that user. A session lasts from the time a user connects until the user disconnects or exits the database application.

Multiple sessions can be created and exist concurrently for a single Oracle database user using the same username. For example, a user with the username/password of HR/HR can connect to the same Oracle Database instance several times.

Instance-Database Configurations



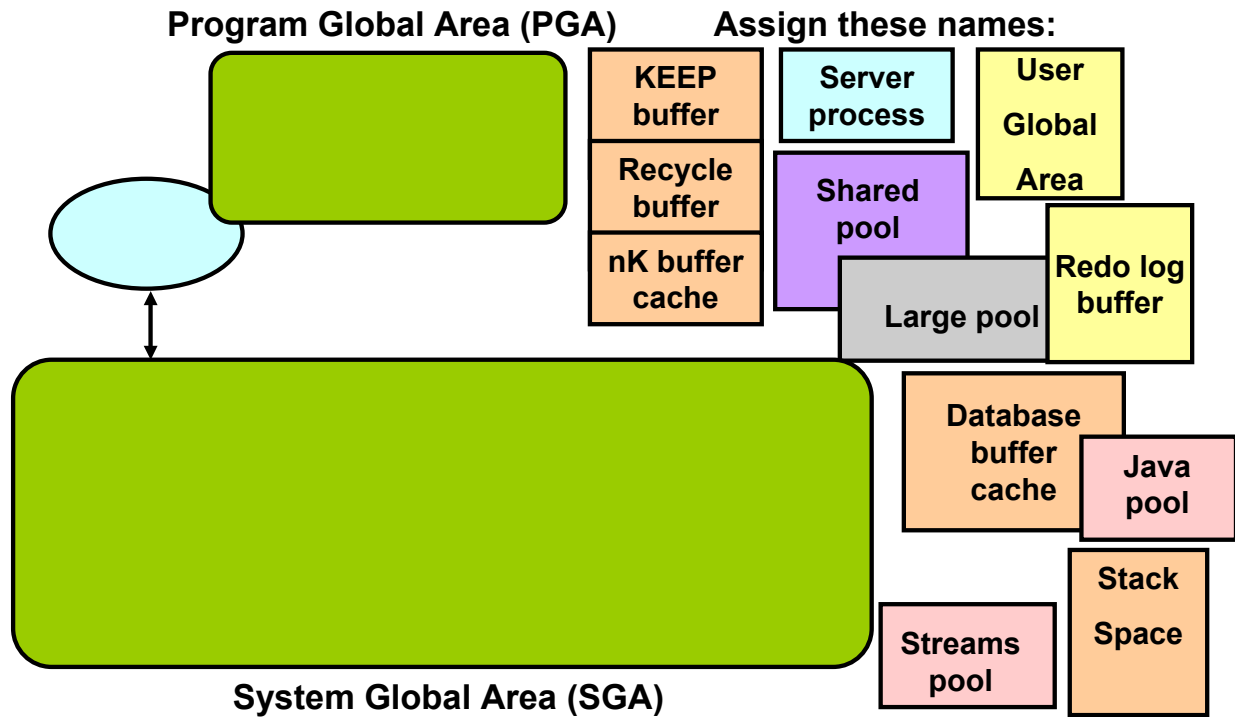
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Instance-Database Configurations

Each database instance is associated with one and only one database. If there are multiple databases on the same server, there is a separate and distinct database instance for each database. A database instance cannot be shared. A Real Application Clusters (RAC) database usually has multiple instances on separate servers for the same shared database. In this model, the same database is associated with each RAC instance, which preserves the requirement that at most only one database be associated with an instance.

Naming the Memory Structures of an Oracle Database



Copyright © 2009, Oracle. All rights reserved.

ORACLE

Oracle University and Al-Khobara for Adaptive Knowledge use only

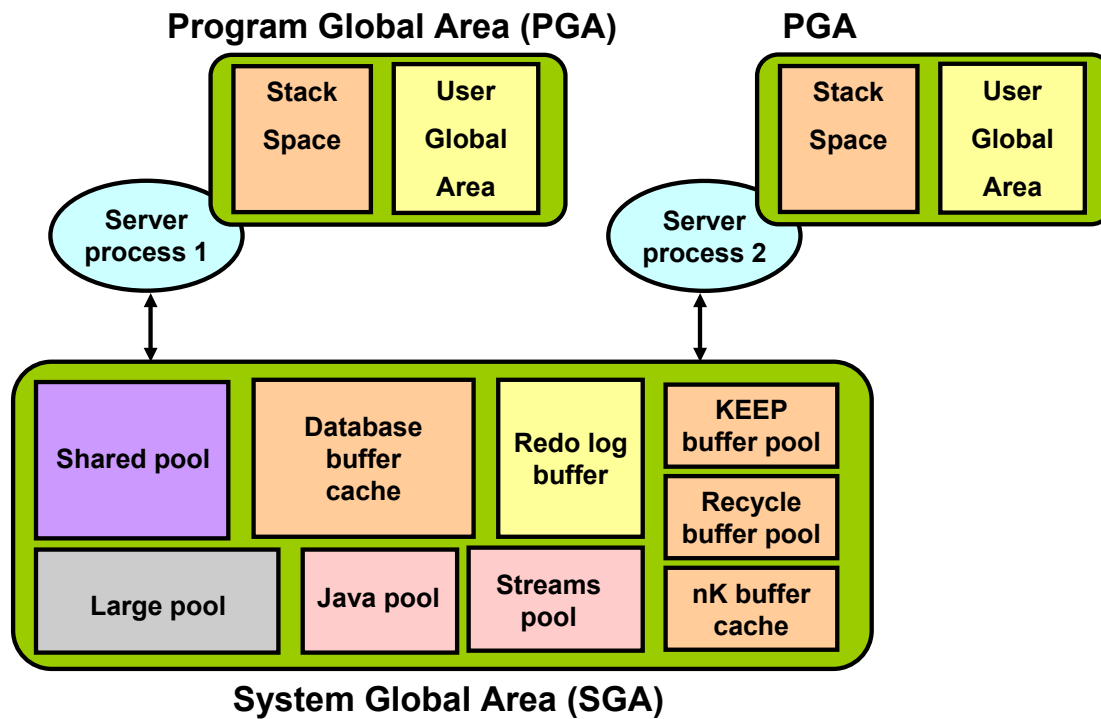
Naming the Memory Structures of an Oracle Database

To get you started:

1. Which are the components of the PGA: _____ and _____.
2. Name the main components of the SGA:

- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____

Oracle Database Memory Structures



Copyright © 2009, Oracle. All rights reserved.

Oracle Database Memory Structures

Oracle Database creates and uses memory structures for various purposes. For example, memory stores program code being run, data that is shared among users, and private data areas for each connected user.

Two basic memory structures are associated with an instance:

- **System Global Area (SGA):** Group of shared memory structures, known as SGA components, that contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.
- **Program Global Areas (PGA):** Memory regions that contain data and control information for a server or background process. A PGA is nonshared memory created by Oracle Database when a server or background process is started. Access to the PGA is exclusive to the server process. Each server process and background process has its own PGA.

Oracle Database Memory Structures (continued)

The SGA is the memory area that contains data and control information for the instance. The SGA includes the following data structures:

- **Shared pool:** Caches various constructs that can be shared among users
- **Database buffer cache:** Caches blocks of data retrieved from the database
- **KEEP buffer pool:** A specialized type of database buffer cache that is tuned to retain blocks of data in memory for long periods of time
- **Recycle buffer pool:** A specialized type of database buffer cache that is tuned to recycle or remove block from memory quickly
- **nK buffer cache:** One of several specialized database buffer caches designed to hold block sizes different than the default database block size
- **Redo log buffer:** Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on the disk
- **Large pool:** Optional area that provides large memory allocations for certain large processes, such as Oracle backup and recovery operations, and I/O server processes
- **Java pool:** Used for all session-specific Java code and data in the Java Virtual Machine (JVM)
- **Streams pool:** Used by Oracle Streams to store information required by capture and apply

When you start the instance by using Enterprise Manager or SQL*Plus, the amount of memory allocated for the SGA is displayed.

A Program Global Area (PGA) is a memory region that contains data and control information for each server process. An Oracle server process services a client's requests. Each server process has its own private PGA that is allocated when the server process is started. Access to the PGA is exclusive to that server process, and the PGA is read and written only by the Oracle code acting on its behalf. The PGA is divided into two major areas: stack space and the user global area (UGA).

With the dynamic SGA infrastructure, the sizes of the database buffer cache, the shared pool, the large pool, the Java pool, and the Streams pool can change without shutting down the instance.

The Oracle database uses initialization parameters to create and manage memory structures. The simplest way to manage memory is to allow the database to automatically manage and tune it for you. To do so (on most platforms), you only have to set a target memory size initialization parameter (MEMORY_TARGET) and a maximum memory size initialization parameter (MEMORY_MAX_TARGET).

Process Architecture

- User process
 - Is the application or tool that connects to the Oracle Database
- Database processes
 - Server process: Connects to the Oracle instance and is started when a user establishes a session
 - Background processes: Are started when an Oracle instance is started
- Daemon/Application processes
 - Networking listeners
 - Grid Infrastructure daemons

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Process Architecture

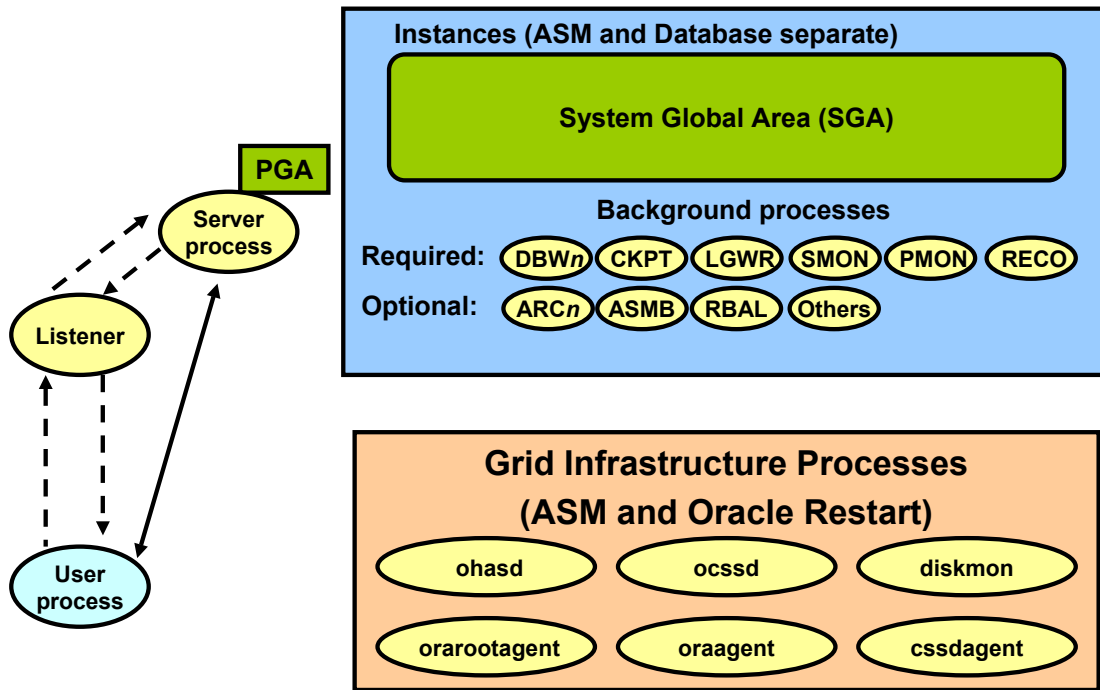
The processes in an Oracle Database system can be divided into three major groups:

- User processes that run the application or Oracle tool code
- Oracle Database processes that run the Oracle database server code (including server processes and background processes)
- Oracle daemons and application processes not specific to a single database

When a user runs an application program or an Oracle tool such as SQL*Plus, the term *user process* is used to refer to the user's application. The user process may or may not be on the database server machine. Oracle Database also creates a *server process* to execute the commands issued by the user process. In addition, the Oracle server also has a set of *background processes* for an instance that interact with each other and with the operating system to manage the memory structures, asynchronously perform I/O to write data to disk, and perform other required tasks. The process structure varies for different Oracle Database configurations, depending on the operating system and the choice of Oracle Database options. The code for connected users can be configured as a dedicated server or a shared server.

- **Dedicated server:** For each session, the database application is run by a user process that is served by a dedicated server process that executes Oracle database server code.
- **Shared server:** Eliminates the need for a dedicated server process for each connection. A dispatcher directs multiple incoming network session requests to a pool of shared server processes. A shared server process serves any client request.

Process Structures



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Process Structures

Server Processes

Oracle Database creates server processes to handle the requests of user processes connected to the instance. The user process represents the application or tool that connects to the Oracle database. It may be on the same machine as the Oracle database or it may exist on a remote client and utilize a network to reach the Oracle database. The user process first communicates with a listener process that creates a server process in a dedicated environment.

Server processes created on behalf of each user's application can perform one or more of the following:

- Parse and run SQL statements issued through the application
- Read necessary data blocks from data files on disk into the shared database buffers of the SGA (if the blocks are not already present in the SGA)
- Return results in such a way that the application can process the information

Background Processes

To maximize performance and accommodate many users, a multiprocess Oracle Database system uses some additional Oracle Database processes called *background processes*. An Oracle Database instance can have many background processes.

Process Structures (continued)

The background processes commonly seen in non-RAC, non-ASM environments can include the following:

- Database writer process (DBW n)
- Log writer process (LGWR)
- Checkpoint process (CKPT)
- System monitor process (SMON)
- Process monitor process (PMON)
- Recoverer process (RECO)
- Job queue coordinator (CJQ0)
- Job slave processes (J nnn)
- Archiver processes (ARC n)
- Queue monitor processes (QMN n)

Other background processes may be found in more advanced configurations such as RAC. See the V\$BGPROCESS view for more information on the background processes.

Some background processes are created automatically when an instance is started, whereas others are started as required.

Other process structures are not specific to a single database, but rather can be shared among many databases on the same server. The Grid Infrastructure and networking processes fall into this category.

Oracle Grid Infrastructure processes on Linux and Unix systems include the following:

- ohasd: Oracle High Availability Service daemon that is responsible to starting Oracle Clusterware processes
- ocssd: Cluster Synchronization Service daemon
- diskmon: Disk Monitor daemon that is responsible for input and output fencing for HP Oracle Exadata Storage Server.
- cssdagent: Starts, stops and check the status of the CSS daemon, ocssd.
- oraagent: Extend clusterware to support Oracle-specific requirements and complex resources
- orarootagent: A specialized Oracle agent process that that helps manage resources owned by root, such as the network.

Adding Process Names

- | | |
|---|-----------------------------------|
| 1. The _____ process writes the dirty buffers to the data files. | A. Checkpoint process (CKPT) |
| 2. The _____ process writes the redo entries to the online redo log files. | B. System monitor process (SMON) |
| 3. The _____ process writes checkpoint information in the control file and each data file header. | C. Recoverer process (RECO) |
| 4. The _____ process performs recovery on instance startup. | D. Log writer process (LGWR) |
| 5. The _____ process performs process recovery when a user process fails. | E. Archiver processes (ARCn) |
| 6. The _____ process resolves in-doubt distributed transactions. | F. Process monitor process (PMON) |
| 7. The _____ processes copy redo log files to a designated storage device. | G. Database writer process (DBWn) |

ORACLE

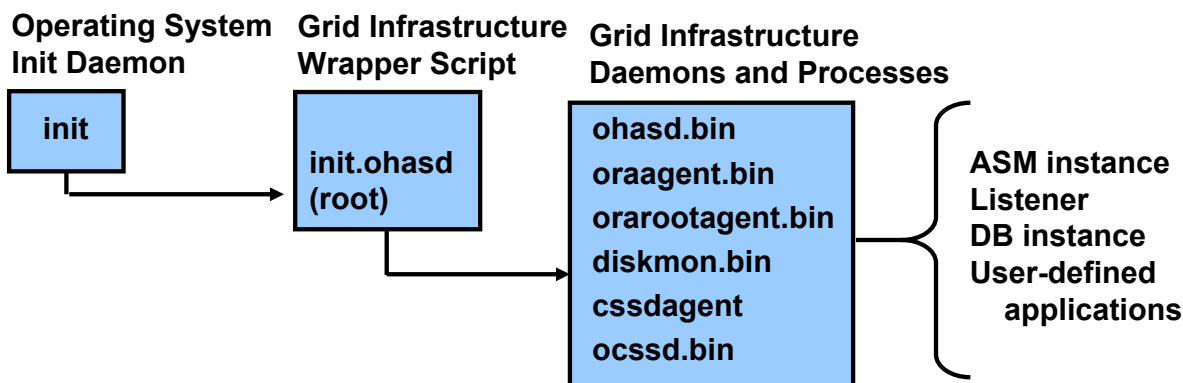
Copyright © 2009, Oracle. All rights reserved.

Adding Process Names

After you fill in the process names, see “Appendix A Solutions” for possible answers.

Process Startup Sequence

- Oracle Grid Infrastructure is started by the OS init daemon.



- Oracle Grid Infrastructure installation modifies the `/etc/inittab` file to ensure startup every time the machine is started in the corresponding run level.

```
# cat /etc/inittab
..
h1:35:respawn:/etc/init.d/init.ohasd run >/dev/null 2>&1 </dev/null
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Process Startup Sequence

During the installation of Oracle Grid Infrastructure, entries are placed in the operating system `/etc/inittab` file to start a wrapper script. The wrapper script is responsible for setting up environment variables and then starting the Oracle Grid Infrastructure daemons and processes.

When a command is used to stop Oracle Grid Infrastructure, the daemons will be stopped, but the wrapper script process will remain running.

The format of the UNIX `/etc/inittab` file is as follows:

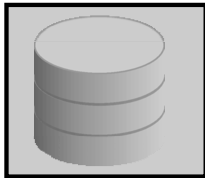
```
id : run levels : action : process with parameters
```

The wrapper script is started with the `respawn` action so it will be restarted whenever it is terminated.

Some of the Oracle Grid Infrastructures daemons will be running under the root user with real time priority, and others will be running under the Grid Infrastructure owner with user-mode priorities after they are started. On a Windows platform, operating system services are used instead of wrapper initialization scripts and the daemons are executable binaries.

Note: It is not supported to execute the wrapper script directly.

Database Storage Architecture



Control files



Data files



Online redo log files



Parameter file



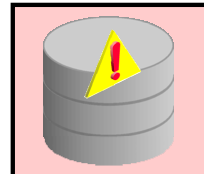
Backup files



Archived redo log files



Password file



Alert log and trace files

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Storage Architecture

The files that constitute an Oracle database are organized into the following:

- **Control files:** Contain data about the database itself (that is, physical database structure information). These files are critical to the database. Without them, you cannot open data files to access the data in the database. It can also contain metadata related to backups.
- **Data files:** Contain the user or application data of the database, as well as metadata and the data dictionary
- **Online redo log files:** Allow for instance recovery of the database. If the database server crashes and does not lose any data files, the instance can recover the database with the information in these files.

The following additional files are important to the successful running of the database:

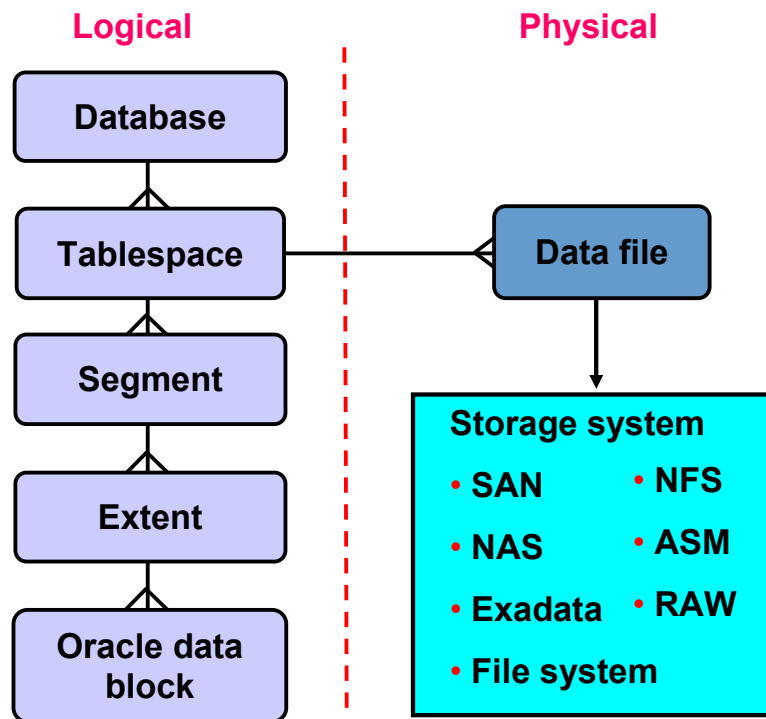
- **Parameter file:** Is used to define how the instance is configured when it starts up
- **Password file:** Allows sysdba, sysoper, and sysasm to connect remotely to the instance and perform administrative tasks
- **Backup files:** Are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.
- **Archived redo log files:** Contain an ongoing history of the data changes (redo) that are generated by the instance. Using these files and a backup of the database, you can recover a lost data file. That is, archive logs enable the recovery of restored data files.

Database Storage Architecture (continued)

- **Trace files:** Each server and background process can write to an associated trace file. When an internal error is detected by a process, the process dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, whereas other information is for Oracle Support Services.
- **Alert log file:** These are special trace entries. The alert log of a database is a chronological log of messages and errors. Oracle recommends that you review the alert log periodically.

Note: Parameter, password, alert, and trace files are covered in other lessons.

Logical and Physical Database Structures



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Logical and Physical Database Structures

The database has logical structures and physical structures.

Databases, Tablespaces, and Data Files

The relationship among databases, tablespaces, and data files is illustrated in the slide. Each database is logically divided into two or more tablespaces. One or more data files are explicitly created for each tablespace to physically store the data of all segments in a tablespace. If it is a TEMPORARY tablespace, it has a temporary file instead of a data file. A tablespace's data file can be physically stored on any supported storage technology.

Tablespaces

A database is divided into logical storage units called *tablespaces*, which group related logical structures or data files together. For example, tablespaces commonly group all of an application's segments to simplify some administrative operations.

Data Blocks

At the finest level of granularity, an Oracle database's data is stored in *data blocks*. One data block corresponds to a specific number of bytes of physical space on the disk. A data block size is specified for each tablespace when it is created. A database uses and allocates free database space in Oracle data blocks.

Logical and Physical Database Structures (continued)

Extents

The next level of logical database space is an *extent*. An extent is a specific number of contiguous Oracle data blocks (obtained in a single allocation) that are used to store a specific type of information. Oracle data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.

Segments

The level of logical database storage above an extent is called a *segment*. A segment is a set of extents allocated for a certain logical structure. For example:

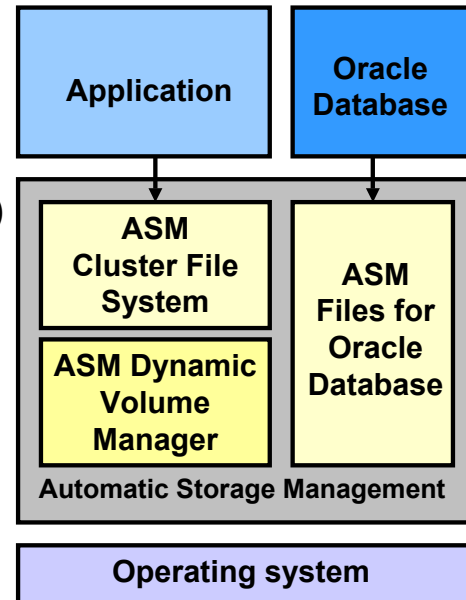
- **Data segments:** Each nonclustered, non-index-organized table has a data segment, with the exception of external tables, global temporary tables, and partitioned tables in which each table has one or more segments. All of the table's data is stored in the extents of its data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- **Index segments:** Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
- **Undo segments:** One UNDO tablespace is created for each database instance. This tablespace contains numerous undo segments to temporarily store undo information. The information in an undo segment is used to generate read-consistent database information and, during database recovery, to roll back uncommitted transactions for users.
- **Temporary segments:** Temporary segments are created by the Oracle database when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the instance for future use. Specify either a default temporary tablespace for every user, or a default temporary tablespace that is used database-wide.

Note: There are other types of segments not listed above. There are also schema objects such as views, packages, triggers, etc. that are not considered segments even though they are database objects. A segment owns its respective disk space allocation. The other objects exist as rows stored in a system metadata segment.

The Oracle database dynamically allocates space. When the existing extents of a segment are full, additional extents are added. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on the disk, and they can come from different data files belonging to the same tablespace.

Automatic Storage Management

- Is a portable and high-performance cluster file system
- Manages Oracle database files
- Manages application files with ASM Cluster File System (ACFS)
- Spreads data across disks to balance load
- Mirrors data in case of failures
- Solves storage-management challenges



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Automatic Storage Management

Automatic Storage Management (ASM) provides vertical integration of the file system and the volume manager for Oracle database files. ASM can provide management for single symmetric multiprocessing (SMP) machines or across multiple nodes of a cluster for Oracle Real Application Clusters (RAC) support.

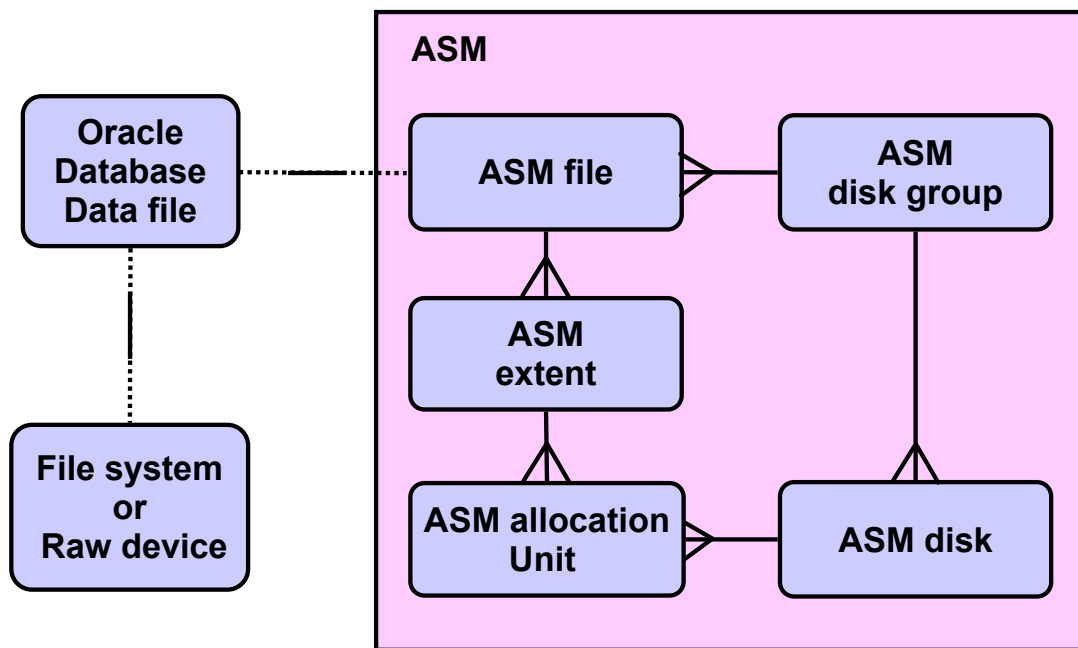
Oracle ASM Cluster File System (ACFS) is a multi-platform, scalable file system, and storage management technology that extends ASM functionality to support application files outside of the Oracle Database such as executables, reports, BFILEs, video, audio, text, images, and other general-purpose application file data.

ASM distributes input/output (I/O) load across all available resources to optimize performance while removing the need for manual I/O tuning. ASM helps DBAs manage a dynamic database environment by enabling them to increase the database size without having to shut down the database to adjust storage allocation.

ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied storage mechanisms. Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per-file basis.

ASM capabilities save the DBA's time by automating manual storage and thereby increasing the administrator's ability to manage more and larger databases with increased efficiency.

ASM Storage Components



ORACLE

Copyright © 2009, Oracle. All rights reserved.

ASM Storage Components

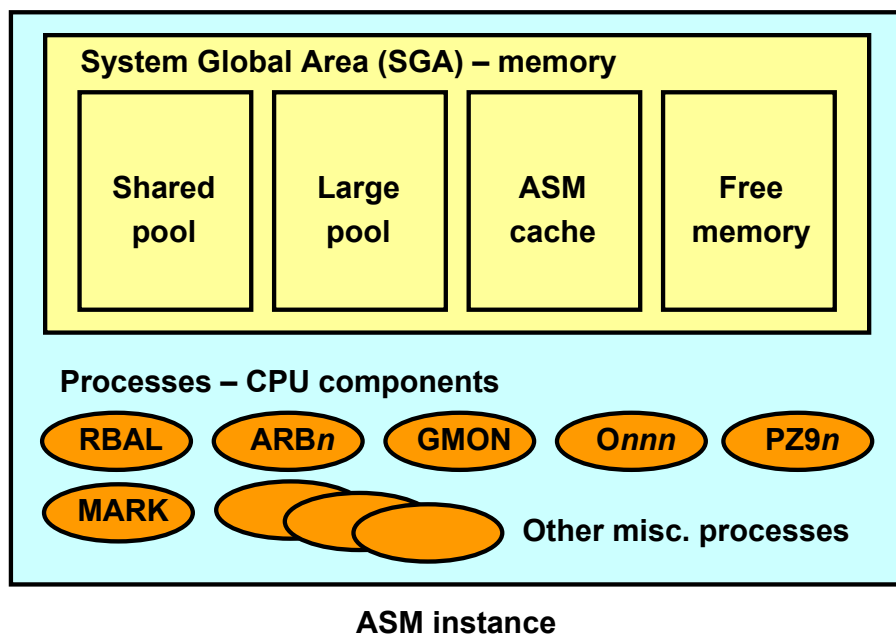
ASM does not eliminate any existing database functionality. Existing databases are able to operate as they always have. New files may be created as ASM files, whereas existing ones are administered in the old way or can be migrated to ASM.

The diagram illustrates the relationships between an Oracle database data file and the ASM storage components. The crow's foot notation represents a one-to-many relationship. An Oracle database data file has a one-to-one relationship with either a file stored on the operating system in a file system or an ASM file.

An Oracle ASM disk group is a collection of one or more Oracle ASM disks managed as a logical unit. The data structures in a disk group are self-contained using some of the space for metadata needs. Oracle ASM disks are the storage devices provisioned to an Oracle ASM disk group and can be physical disk or partitions, a Logical Unit Number (LUN) from a storage array, a logical volume (LV), or a network-attached file. Each ASM disk is divided into many ASM allocation units, the smallest contiguous amount of disk space that ASM allocates. When you create an ASM disk group, you can set the ASM allocation unit size to 1, 2, 4, 8, 16, 32, or 64 MB depending on the disk group compatibility level. One or more ASM allocation units forms an ASM extent. An Oracle ASM extent is the raw storage used to hold the contents of an Oracle ASM file. An Oracle ASM file consists of one or more file extents. Variable extent sizes of 1*AU size, 4*AU size, and 16*AU size are used for supporting very large ASM files.

ASM Instance

The ASM Instance is the process and memory components for ASM.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

ASM Instance

Every time ASM or a database is started, a shared memory area called the system global area (SGA) is allocated and Oracle ASM or database background processes are started. The combination of the background processes and the SGA is called an Oracle ASM instance or an Oracle Database instance. The instance represents the CPU and RAM components of a running ASM environment.

The SGA in an ASM instance is different in memory allocation and usage than the SGA in a database instance. The SGA in the ASM instance is divided into four primary areas as follows:

- **Shared Pool:** Used for metadata information
- **Large Pool:** Used for parallel operations
- **ASM Cache:** Used for reading and writing blocks during rebalance operations
- **Free Memory:** Unallocated memory available

The minimum recommended amount of memory for an ASM instance is 256 MB. Automatic memory management is enabled by default on an ASM instance and will dynamically tune the sizes of the individual SGA memory components. The amount of memory that is needed for an ASM instance will depend on the amount of disk space being managed by ASM.

The second part of the ASM instance is the background processes. An ASM instance can have many background processes; not all are always present.

ASM Components: ASM Instance (continued)

The background processes specific to ASM functionality are covered in the next slide. There are required background processes and optional background processes. Some of these processes may include the following:

- **ARCn:** The archiver processes
- **CKPT:** The checkpoint process
- **DBWn:** The database writer processes
- **DIAG:** The diagnosability process
- **Jnnn:** Job queue processes
- **LGWR:** The log writer process
- **PMON:** The process monitor process
- **PSP0:** The process spawner process
- **QMNn:** The queue monitor processes
- **RECO:** The recoverer process
- **SMON:** The system monitor process
- **VKTM:** The virtual keeper of time process
- **MMAN:** The memory manager process

The above list of processes is not complete. For the ASM instance, these processes will not always perform the same tasks as they would in a database instance. For example, the LGWR process in a database instance is responsible for copying change vectors from the log buffer section of the SGA to the online redo logs on disk. The ASM instance does not contain a log buffer in its SGA, nor does it use online redo logs. The LGWR process in an ASM instance copies logging information to an ASM disk group.

If ASM is clustered, then additional processes related to cluster management will be running in the ASM instance. Some of these processes include the following:

- **LMON:** The global enqueue service monitor process
- **LMDn:** The global enqueue service daemons
- **LMSn:** The global cache service processes
- **LCKn:** The lock processes

For more about managing the ASM instance, see Appendix D and the documentation.

DBA Configuration Tools

Setting up the technical environment for this course included the following tasks and tools:

- Installing and configuring the Oracle Grid Infrastructure for a stand-alone server with the OUI, including:
 - Configuring a listener
 - Creating an ASM instance (+ASM) and configuring the +DATA disk group
 - Configuring Oracle Restart
- Creating and configuring additional ASM disk groups (such as +FRA) with `asmca`
- Installing the Oracle Database 11g software with OUI
- Creating the `orcl` Oracle database with `dbca`

Note: These tasks have already been performed for you.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

DBA Configuration Tools

There are several configuration tools available to assist DBAs with the configuration and management of their Oracle environment. Some of these tools include:

- **Oracle Universal Installer (OUI):** Installs your Oracle software and options. Depending on the product being installed and the options specified during installation, the OUI can automatically call other tools to perform additional tasks, such as the creation of an Oracle database.
- **Oracle ASM Configuration Assistant (ASMCA):** Supports installing and configuring ASM instances, disk groups, volumes, and Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- **Oracle Database Configuration Assistant (DBCA):** Allows you to create and delete Oracle databases as well as modify database options and manage database templates. This utility is called from the OUI during the installation of the Oracle database software if you chose to create a database as part of that installation.
- **Net Configuration Assistant (NETCA):** Configures listeners and naming methods, which are critical components of an Oracle environment

DBA Configuration Tools (continued)

Note: Oracle Restart is installed and configured as part of an Oracle Grid Infrastructure for stand-alone server installation. Oracle Restart manages Oracle component dependencies and automatically restarts the various Oracle components after a hardware or software failure or whenever your database host computer restarts. If the Oracle Grid Infrastructure installation is done before the installation of the Oracle database software, any databases created will be automatically configured with Oracle Restart.

Management Framework and Related DBA Tools

The Oracle database management framework includes:

- Database instance
- Listener
- Management interface:
 - Management agent (when using Grid Control)
 - Database Control

Related tools and commands include:
SQL*Plus:

```
SQL> startup
SQL> shutdown immediate
```

Listener Control utility:

```
$ lsnrctl status
```

Enterprise Manager Control utility:

```
$ emctl status dbconsole
$ emctl start dbconsole
$ emctl stop dbconsole
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Management Framework and Related DBA Tools

There are three major components of the Oracle database management framework:

- The database instance that is being managed
- A listener that allows connections to the database
- The management interface. This may be either a management agent running on the node where the database server runs (which connects it to Oracle Enterprise Manager Grid Control) or the stand-alone Oracle Enterprise Manager Database Control. This is also referred to as the *Database Console*.

Related tools and commands to start, stop, or check the status are:

- For the database instance, in SQL*Plus: `startup` and `shutdown immediate`
- For the listener: `lsnrctl status`
- For Enterprise Manager Database Control:
 - `emctl dbconsole status`
 - `emctl dbconsole start`
 - `emctl dbconsole stop`

Facilitating Database Management with Oracle Restart

- Restarting Oracle components when the host computer restarts or after hardware or software failure
- Monitoring components and restarting them, if needed
- For single-instance environments
- Considering component dependencies:
 - Mounting disk groups and starting the ASM instance before starting the database instance
 - Soft dependency between the database instance and the listener
- Starting Oracle Restart with the `crsctl` utility
- Managing Oracle Restart components with the `srvctl` utility

```
$ srvctl stop database -d orcl -o abort
```



Copyright © 2009, Oracle. All rights reserved.

Facilitating Database Management with Oracle Restart

- With Oracle Restart, the various Oracle components are automatically restarted after a hardware or software failure or whenever your database host computer restarts.
- Oracle Restart performs periodic checks to monitor the health of these components. If a check operation fails for a component, the component is shut down and restarted.
- Oracle Restart is used in single-instance (nonclustered) environments only. For Oracle Real Application Clusters (Oracle RAC) environments, the functionality to automatically restart components is provided by Oracle Clusterware.
- Oracle Restart ensures that Oracle components are started in the proper order, considering component dependencies. For example, if database files are stored in ASM disk groups, then before starting the database instance, Oracle Restart ensures that the ASM instance is started and the required disk groups are mounted. Likewise, if a component must be shut down, Oracle Restart ensures that dependent components are cleanly shut down first.
- Oracle Restart also manages the soft dependency between database instances and the Oracle Net listener (the listener). When a database instance is started, Oracle Restart attempts to start the listener. If the listener startup fails, the database is still started. If the listener later fails, Oracle Restart does not shut down and restart any database instances.
- You start Oracle Restart with the Clusterware Control (`crsctl`) utility.
- Oracle Restart includes the Server Control (`srvctl`) utility that you use to start and stop Oracle Restart-managed components.

Facilitating Database Management with Oracle Restart (continued)

Note: The `srvctl` utility is located in both the `$ORACLE_HOME/bin` directory for the Grid Infrastructure software and the `$ORACLE_HOME/bin` directory for the Oracle database software. You should use the `srvctl` utility from the Oracle database software when starting the Oracle database. You should use the `srvctl` utility from the Grid Infrastructure software when starting the ASM instance or the listener.

Quiz

Oracle Restart is installed and configured as part of an Oracle Grid Infrastructure for stand-alone server installation.

1. True
2. False

ORACLE

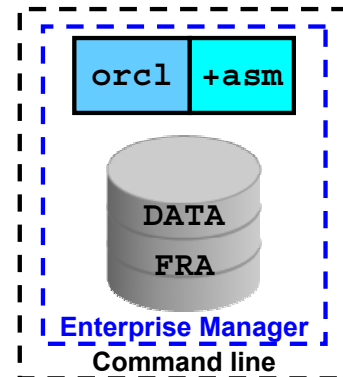
Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Summary

In this lesson, you should have learned how to:

- Describe the core concepts of the Oracle Database architecture with ASM
- Determine which DBA configuration and management tools to use for which task
- Describe the technical course environment



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Summary

- The diagram in the slide displays the `orcl` and `+asm` instances as rectangles above a database cylinder, which lists the `DATA` and `FRA` disk groups.
- They are surrounded by a blue dashed line, denoting Enterprise Manager as a possible graphical interface.
- All this is surrounded again by a dashed black line, denoting the command line, as a possible character-based interface.

2

Configuring for Recoverability

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Invoke and configure Recovery Manager (RMAN)
- Configure your database in `ARCHIVELOG` mode
- Configure multiple archive log file destinations to increase availability
- Configure the Fast Recovery Area (FRA)
- Specify a retention policy

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

The Fast Recovery Area (FRA) was formerly known as Flash Recovery Area.

Purpose of Backup and Recovery Functionality

Backup and recovery functionality is needed for the following:

- Data protection
 - Media failure
 - User errors
 - Application errors
- Data preservation and historical retention
- Data transfer

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Purpose of Backup and Recovery Functionality

The purpose of backup and recovery is to restore a failed database. Backups protect the database against problems such as hardware failure, media failure, user errors, and application errors. Media errors cause data problems by failing at the hardware level; a bad controller or disk drive can introduce either subtle or obvious errors. Users can also cause data errors, simply by issuing commands that should not be issued. Those same types of errors can be caused by an application with a bug.

Backups can also be used for data preservation and historical retention. Backups taken and preserved in ARCHIVELOG mode can be used to recover a database to a past point in time, which can be useful to meet compliance regulations.

You can also use backup and recovery tools to move data to other databases—even in other locations. A backup of a database is one possible way to duplicate it at another location.

Typical Backup and Recovery Tasks

To be able to recover from data loss problems with minimal down time:

- Configure the database for recoverability
- Define a backup schedule
- Plan and test different types of failure scenarios
- Monitor, tune, and troubleshoot the backup and recovery environment
- Restore data from backups
- Recover transactions to a desired point in time

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Typical Backup and Recovery Tasks

A robust backup and recovery plan is important for a database that you cannot afford to lose. The plan includes the following tasks:

- **Configuration:** A backup and recovery strategy needs to be configured for your environment. This should include backup method, destination for backups, backup retention time and deletion of backups, and backup protection (encryption), if needed.
- **Scheduling:** Backups should be scheduled to run automatically during nonpeak hours.
- **Testing:** Periodically test backups and recovery practices.
- **Monitoring:** Monitor the effects of the backup operations to determine performance degradation on production databases and improve backup efficiency, when necessary.
- **Restoration:** A corrupted data file is overwritten from a backup of the data file. The data file is at a prior point of time than the current database.
- **Recovery:** Recovery applies the changes to the individual blocks, using archive and redo information, to move the database forward to the current point in time.

Oracle Backup and Recovery Solutions

For a recoverable system:

- RMAN
 - Block media recovery
 - Unused block compression
 - Binary compression
 - Backup encryption
- Solutions achieved via backup types:
 - All data blocks within your chosen files (full or incremental level 0)
 - Only information that has changed since a previous backup (incremental)
 - Cumulative (changes up to last level 0)
 - Differential (changes up to last incremental)



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Backup and Recovery Solutions

The following are major backup and recovery solutions:

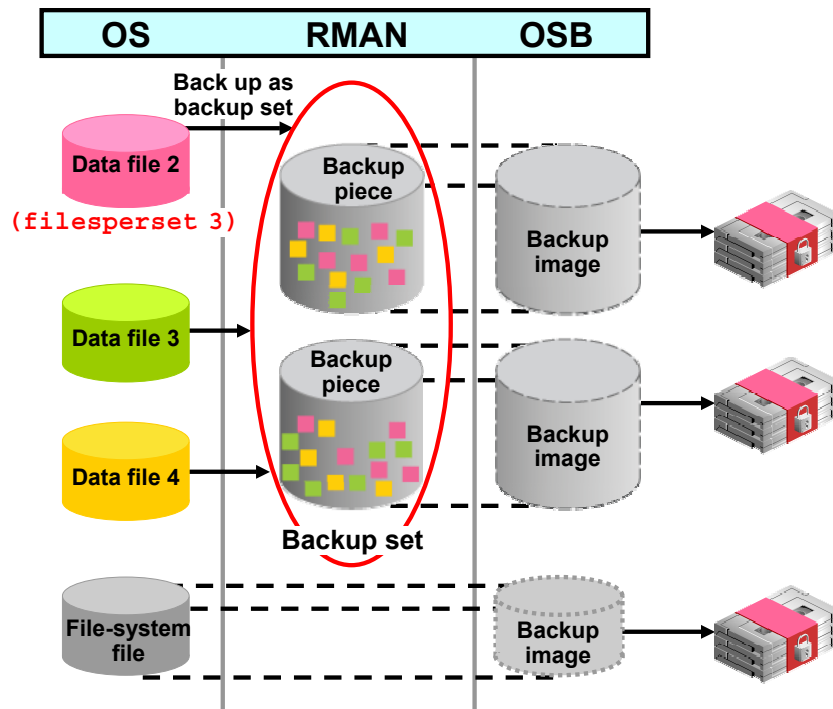
Recovery Manager: A utility (with graphic and command-line interfaces) for performing backup and recovery. Some of the major features available when using RMAN are:

- **Block media recovery:** A method of recovering specific blocks of data, as opposed to entire tables (with Data Pump) or data files (with RMAN)
- **Unused block compression:** Unused block compression is a method by which blocks that are not currently used by the database are not read by the backup, and thus, not included in the backup.
- **Binary compression:** A space-saving feature in which backup files are compressed using well-known algorithms (comparable to utilities such as zip on Linux)
- **Backup encryption:** A security device for protecting backups you make

The solutions are achieved with these backup types:

- **Full backup:** Makes a copy of each data block that contains data and that is within the files being backed up
- **Incremental backup:** Makes a copy of all data blocks that have changed since a previous backup. The Oracle database supports two levels of incremental backup (0 and 1). A level 1 incremental backup can be one of two types: *cumulative* or *differential*. A cumulative backup backs up all changes since the last level 0 backup. A differential backup backs up all changes since the last incremental backup (which could be either a level 0 or level 1 backup).

Oracle Backup Solutions



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Backup Solutions

The slide shows backup sets. On the left are data files at the OS level. You see how they relate to the RMAN image copies and backup pieces (middle section) and how these relate to the Oracle Secure Backup (OSB) backup images.

At the bottom of the diagram are file-system files, which do not have an RMAN equivalent; they relate directly to OSB backup images.

- RMAN backs up data files, control files, redo log archives, and SPFILEs, whether they are the originals, backup sets, or image copies. RMAN performs backup and recovery operations to disk, and with the help of a media management layer (MML) such as Oracle Secure Backup, also to and from tape.
- Oracle Secure Backup (OSB) is a centralized tape management software for your entire Oracle environment, including file systems and the Oracle database. OSB can back up and restore data locally or over the local area network (LAN), wide area network (WAN), or SAN.

This diagram shows a subset of the Oracle Backup solution. Not included are image copies and proxy copies, which may be supported by some media managers.

Terminology Review

Match the following terms and descriptions:

1. ____ backs up a portion of the database. The control file may or may not be included.
2. ____ is a consistent backup because the SCN in data file headers matches the SCN in the control files.
3. ____ backs up each data block that contains data and that is within the files being backed up.
4. ____ is an inconsistent backup because there is no guarantee that the data files are synchronized with the control files.
5. ____ includes all data files and at least one control file.

(W) Whole database backup (F) Full backup (C) Cold or offline backup (P) Partial database backup (O) Online backup

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Terminology Review

Correct Answers: 1P, 2C, 3F, 4O, 5W

Terminology Review

Which description fits best the following backup types:

1. Image copies
2. Backup sets

Description:

- A) They are collections of one or more binary files that contain one or more data files, control files, server parameter files, or archived log files. Empty data blocks and currently unused blocks are not stored.
- B) They are duplicates of data or archived log files (similar to a file copy).

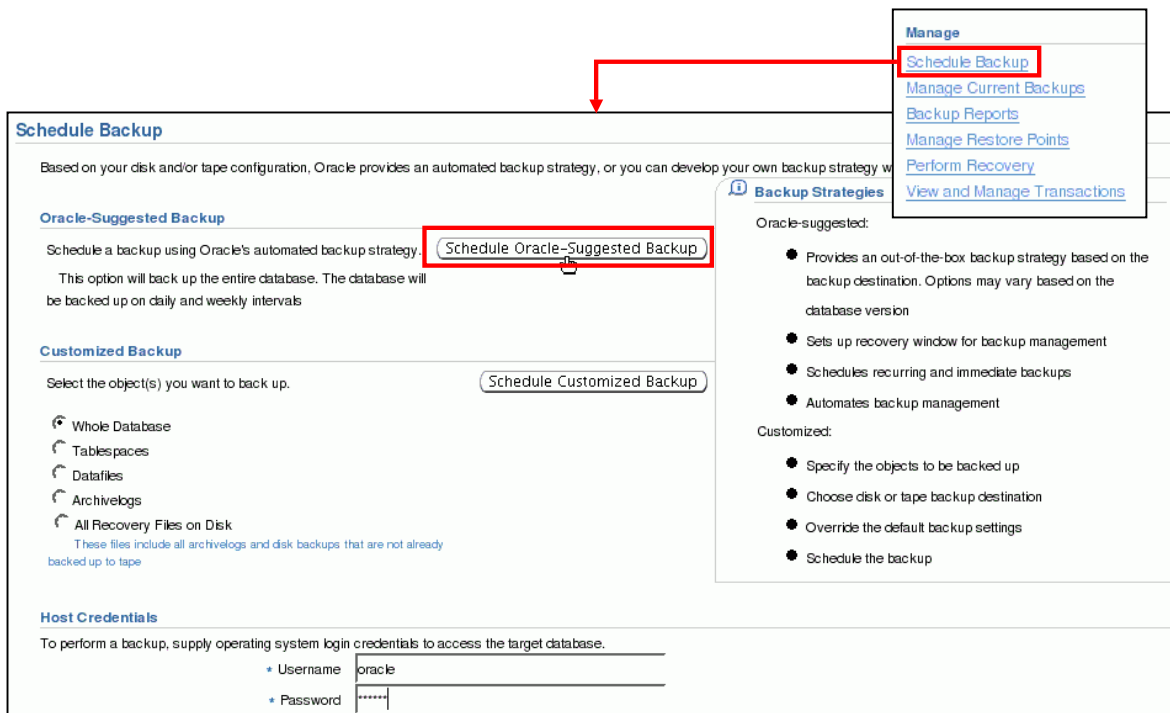
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Terminology Review (continued)

Correct Answers: 1B, 2A

What You Already Know: Oracle-Suggested Backup



ORACLE

Copyright © 2009, Oracle. All rights reserved.

What You Already Know: Oracle-Suggested Backup

Enterprise Manager makes it easy for you to set up an Oracle-suggested backup strategy that protects your data and provides efficient recoverability to any point in the preceding 24 hours, and possibly as far back as 48 hours, depending on when the last backup was created. The Oracle-suggested strategy uses the incremental backup and incrementally updated backup features, providing faster recoverability than is possible when applying database changes from the archived log files.

To establish an Oracle-suggested strategy, navigate to the Maintenance page. In the Backup/Recovery region, select Schedule Backup. The Backup Strategies section enables you to select from the Oracle-suggested backup and Customized backup strategies. The Oracle-suggested strategy takes a full database copy as the first backup. Because it is a whole database backup, you might want to consider taking this at a period of least activity. After that, an incremental backup to disk is taken every day. Optionally, a weekly tape backup can be made, which backs up all recovery-related files.

Because these backups on disk are retained, you can always perform a full database recovery or a point-in-time recovery to any time within the past 24 hours, at the minimum. The recovery time could reach back as far as 48 hours. This is because just before a backup is taken on a given day, the backup from the *beginning* of day $n-1$ still exists.

Using Recovery Manager

```
$ rman target /  
  
RMAN> BACKUP DATABASE;  
Starting backup at 10-JUN-07  
.  
.  
RMAN> LIST BACKUP;  
BS Key   Type LV Size       Device Type Elapsed Time Completion Time  
-----  
1        Full  1.06G    DISK          00:01:49      10-JUN-07  
.  
.  
RMAN> DELETE OBSOLETE;  
.  
.  
Do you really want to delete the above objects (enter YES or NO)? YES  
deleted archived log  
.  
.
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Recovery Manager

Invoke RMAN at the operating system command line and specify the appropriate options. The following are the most commonly used options:

- **target:** The connect-string for the target database
- **catalog:** The connect-string for a recovery catalog
- **nocatalog:** Specifies there is no recovery catalog. This is the default.
- **cmdfile:** The name of an input command file
- **log:** The name of the output message log file

The RMAN invocation shown in the slide simply connects to the local database, as the target.

Here is an example of an RMAN invocation that connects to the local database using OS authentication, and specifies a command file to be run and a log file to receive a transcript of the RMAN commands that belong to the session:

```
$ rman target / cmdfile=~/.fullbu.rman log=~/.fullbu.log
```

At the RMAN prompt, you can submit RMAN commands to manage your backup environment and create backups in many different ways, depending on your needs. Shown in the slide are commands to list the existing backups (LIST BACKUP) and delete any obsolete backups (DELETE OBSOLETE). The specifics of what these and other commands do are covered throughout this course.

Note: Refer to the *Oracle Database Backup and Recovery User's Guide* for more information about how to invoke RMAN. Refer to the *Oracle Database Backup and Recovery Reference* for the

complete list of RMAN commands and their options.

Types of RMAN Commands

RMAN commands are of the following types:

- Stand-alone command:
 - Is executed individually at the RMAN prompt
 - Cannot appear as subcommands within `RUN`
- Job command:
 - Must be within the braces of a `RUN` command
 - Is executed as a group

Some commands can be executed as either a stand-alone or a job command.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Types of RMAN Commands

You can issue two basic types of RMAN commands: stand-alone and job commands.

Stand-alone commands are executed at the RMAN prompt and are generally self-contained. Some of the stand-alone commands are:

- `CHANGE`
- `CONNECT`
- `CREATE CATALOG, RESYNC CATALOG`
- `CREATE SCRIPT, DELETE SCRIPT, REPLACE SCRIPT`

Job commands are usually grouped and executed sequentially inside a command block. If any command within the block fails, RMAN ceases processing; no further commands within the block are executed. The effects of any already executed commands still remain, though; they are not undone in any way.

An example of a command that can be run only as a job command is `ALLOCATE CHANNEL`. The channel is allocated only for the execution of the job, so it cannot be issued as a stand-alone command. There are some commands that can be issued either at the prompt or within a `RUN` command block, such as `BACKUP DATABASE`. If you issue stand-alone commands, RMAN allocates any needed channels by using the automatic channel allocation feature.

You can execute stand-alone and job commands in interactive mode or batch mode.

Job Commands: Example

Job commands appear inside a RUN command block:

```
RMAN> RUN
2> {
3>   ALLOCATE CHANNEL c1 DEVICE TYPE DISK
4>   FORMAT "/disk2/%U";
5>   BACKUP AS BACKUPSET DATABASE;
6>   SQL 'alter system archive log current';
7> }
```

Execution of the entire block starts
when this line is entered.

Deallocated after the
RUN block completes

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Job Commands: Example

Unlike stand-alone commands, job commands must appear within the braces of a RUN command. Commands placed inside a RUN block as shown in the slide are run as a single unit of commands. Any configurations made within the run block apply within the scope of the block and override any previously made settings. The following are examples of job commands, which must appear inside a RUN block:

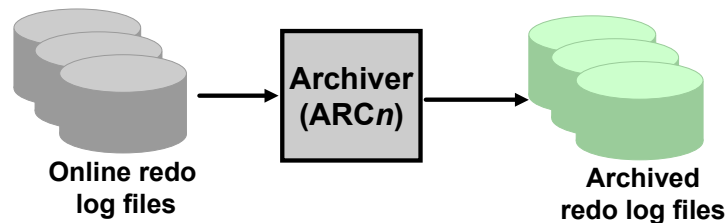
- ALLOCATE CHANNEL
- SWITCH

RMAN executes the job commands inside a RUN command block sequentially. If any command within the block fails, then RMAN ceases processing; no further commands within the block are executed. In effect, the RUN command defines a unit of command execution. When the last command within a RUN block completes, the Oracle database releases any server-side resources such as input/output (I/O) buffers or I/O slave processes allocated within the block.

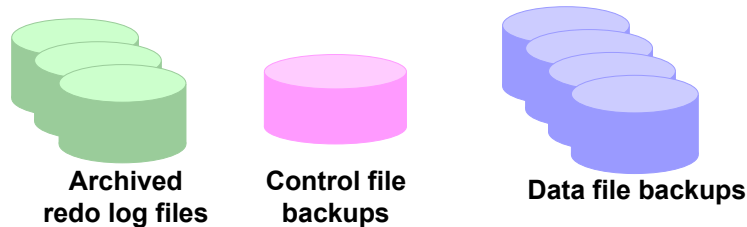
Note: The SQL command on line 6 is just an example. It is NOT a required command for the backup operation.

Configuring Your Database for Backup and Recovery Operations

- Operate the database in ARCHIVELOG mode.



- Configure the FRA.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring Your Database for Backup and Recovery Operations

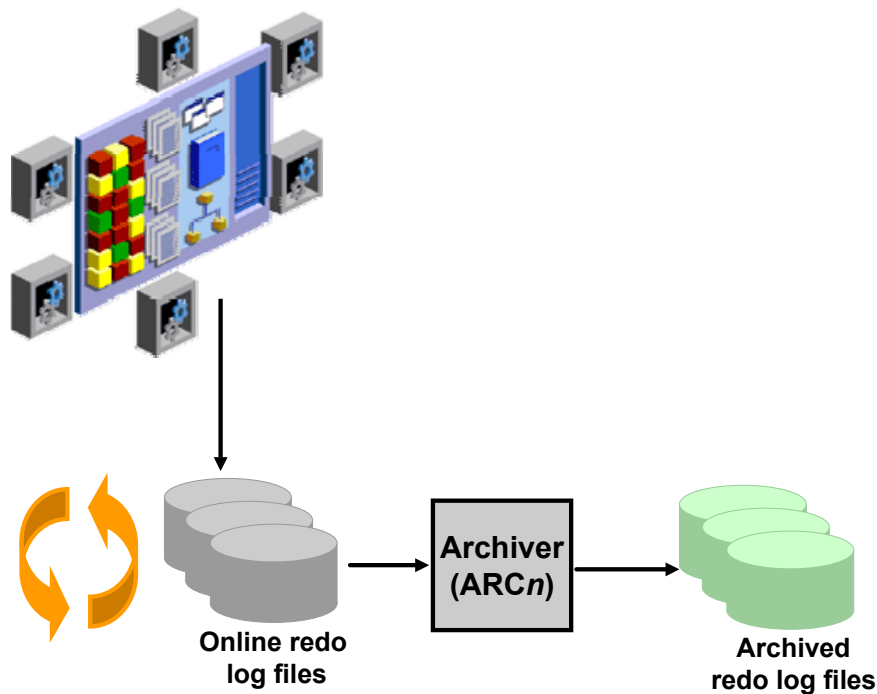
When you operate your database in ARCHIVELOG mode, you have more recovery options after a data loss, including point-in-time recovery of the database or some tablespaces.

It is recommended that you take advantage of the Fast Recovery Area to store as many backup and recovery-related files as possible, including disk backups and archived redo logs.

Some features of Oracle Database backup and recovery, such as Oracle Flashback Database and guaranteed restore points, require the use of a Fast Recovery Area.

Both of the features are covered in detail later in this lesson.

ARCHIVELOG Mode



ORACLE

Copyright © 2009, Oracle. All rights reserved.

ARCHIVELOG Mode

As modifications to data in the database are made, the redo data is written out to the online redo log file. A given file is specified as being written to at a given time. When it is full, the Archiver process (ARCn) copies the online log file to another location that serves as an archive of that file, which can be preserved for as long as you need it. This provides more opportunities for recovery, because you can save, back up, and restore all of the archive redo logs ever generated.

Because the online redo log files are reused in a circular fashion, there is a protocol for controlling when one is allowed to be reused. In ARCHIVELOG mode, the database only begins writing to an online redo log file if it has been archived. This ensures that every redo log file has a chance to be archived.

Configuring ARCHIVELOG Mode

To place the database in ARCHIVELOG mode, perform the following steps:

- Using Enterprise Manager
 - Select the “ARCHIVELOG Mode” check box.
 - Click Apply. The database can be set to ARCHIVELOG mode only from the MOUNT state.
 - Click Yes when asked whether you want to restart the database.
- Using SQL commands
 - Mount the database.
 - Issue the ALTER DATABASE ARCHIVELOG command.
 - Open the database.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring ARCHIVELOG Mode

Placing the database in ARCHIVELOG mode prevents redo logs from being overwritten until they have been archived.

In Enterprise Manager, do this by navigating to Availability > Recovery Settings and selecting the ARCHIVELOG Mode check box. The database must be restarted after making this change.

To issue the SQL command to put the database in ARCHIVELOG mode, the database must be in MOUNT mode. If the database is currently open, you must shut it down cleanly (not abort), and then mount it. The following shows the commands to shut down an open database, put it in ARCHIVELOG mode, and then open it:

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP MOUNT
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER DATABASE OPEN;
```

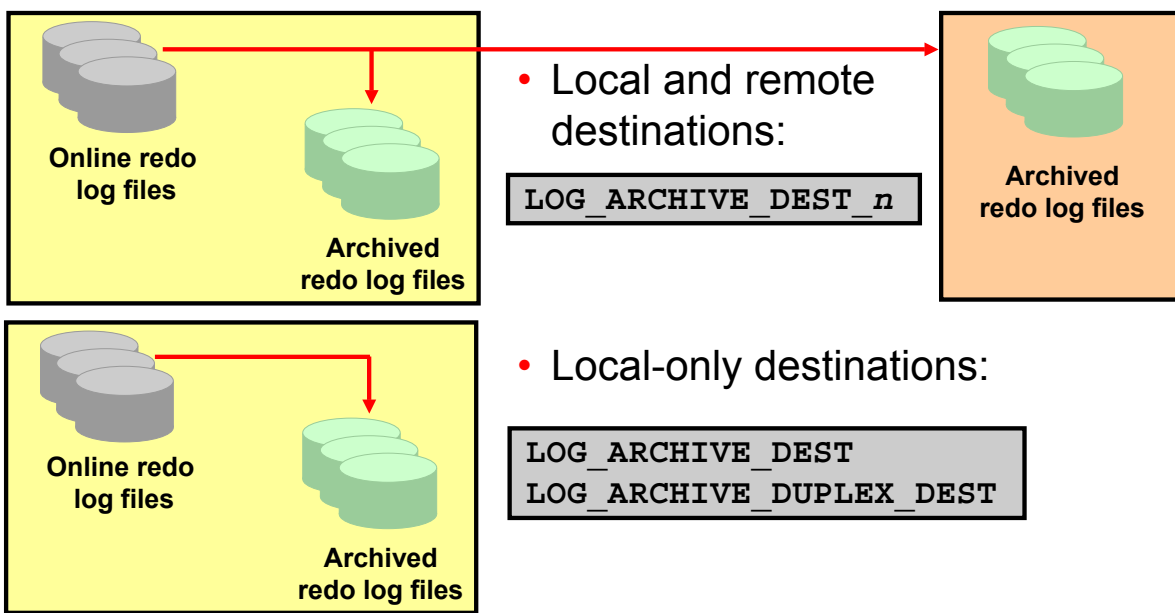
With the database in NOARCHIVELOG mode (the default), recovery is possible only until the time of the last backup. All transactions made after that backup are lost.

In ARCHIVELOG mode, recovery is possible until the time of the last commit. Most production databases are operated in ARCHIVELOG mode.

Note: Back up your database after switching to ARCHIVELOG mode because your database is recoverable only from the first backup taken in that mode.

Configuring Archive Log Destinations

Best practice tip: Create multiple destinations. If you had only one and it would fill up, the database would stop.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring Archive Log Destinations

Best practice tip: You should create multiple archive log destinations, because if you had only one destination and it would fill up, then your database would stop.

Local and remote destinations: Specify local and remote destinations by setting the set of `LOG_ARCHIVE_DEST_n` initialization parameters. There are ten of these, so n can be 1 through 10.

- In order to specify a local storage location, supply a local directory name for the value of one of these variables, supplying the "LOCATION=" string. For example, to specify the `/disk3/arch` directory, set one of these variables as follows:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=/disk3/arch'
```

- If you want to specify a remote location for a standby database, use the `SERVICE` keyword in the value, as in the following example, where `standby1` is the network service name for the standby database instance:

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1'
```

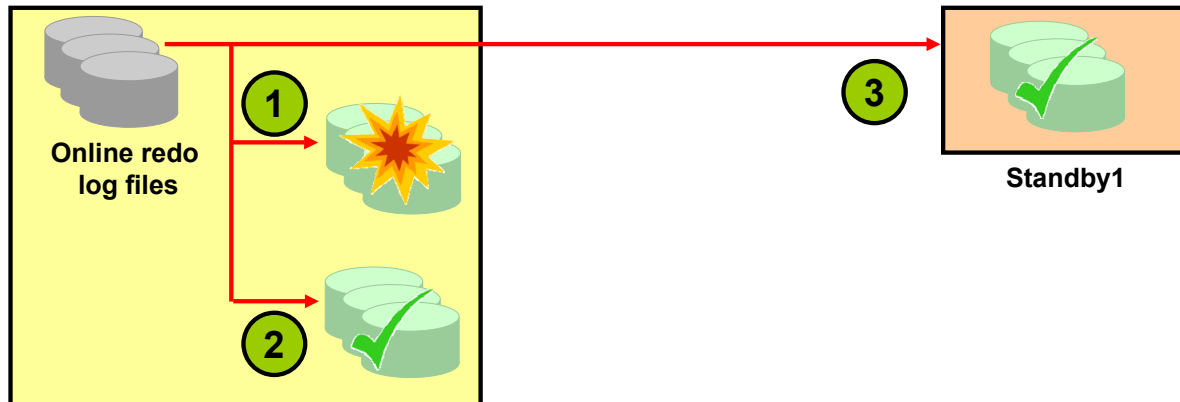
Local-only destinations: With the Standard edition of the Oracle database, set the `LOG_ARCHIVE_DEST` and the `LOG_ARCHIVE_DUPLEX_DEST` parameters to local disk directories. Thus, you can have up to two archive log file locations. For example:

```
LOG_ARCHIVE_DEST = '/disk1/arch'
```

```
LOG_ARCHIVE_DUPLEX_DEST = '/disk2/arch'
```

Where available, Oracle recommends that you use the `LOG_ARCHIVE_DEST_n` method, because this allows the most flexibility in the type and the number of destinations.

Guaranteeing Archive Log Success



`LOG_ARCHIVE_MIN_SUCCEED_DEST = 2`

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Guaranteeing Archive Log Success

If you have specified more than one destination for archive log files, you should specify a minimum number of them that are required to succeed in order for the archive to be considered successful. Do this using the `LOG_ARCHIVE_MIN_SUCCEED_DEST` initialization parameter. Set it to the number of destinations that must succeed in receiving the archived log file. The online log file is not reused until this number is met.

In the example in the slide, there are three destinations specified: two are local, and one is remote. `LOG_ARCHIVE_MIN_SUCCEED_DEST` is set to 2, which means as long as at least two of the destinations succeed, the online redo log file can be overwritten. The example shows that destination 1 has failed. That does not stop the database, because two of them succeeded.

You can use this parameter with either of the models described in the previous slide. If you use it with the `LOG_ARCHIVE_DEST_n` model, then this parameter can have values ranging from 1 through 10. If you use it with the `LOG_ARCHIVE_DEST` model, then the values can be 1 or 2, because you can specify only two destinations in that case.

Guaranteeing Archive Log Success (continued)

Specifying MANDATORY and OPTIONAL

When you define a destination, you can specify that it is a mandatory one. Do this by specifying the MANDATORY or OPTIONAL keyword after the location specification. Here is an example:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=/disk3/arch MANDATORY'
```

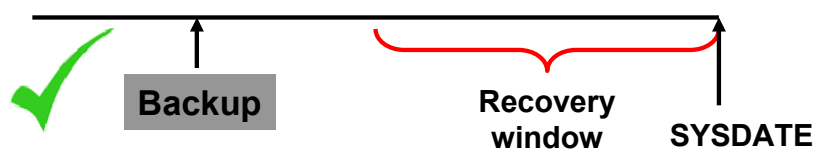
The default is OPTIONAL.

A mandatory destination is given special consideration. If any mandatory destination fails, then Oracle Database considers the archiving of the log to have not succeeded, and the online redo log file is not allowed to be overwritten. In this case, it ignores the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter.

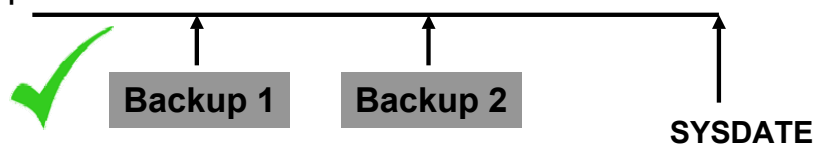
Any destination specified by LOG_ARCHIVE_DEST is mandatory. Any destination declared by LOG_ARCHIVE_DUPLEX_DEST is optional if LOG_ARCHIVE_MIN_SUCCEED_DEST = 1 and mandatory if LOG_ARCHIVE_MIN_SUCCEED_DEST = 2.

Specifying a Retention Policy

- Retention policy: Describes which backups will be kept and for how long
- Two types of retention policies:
 - Recovery window:** Establishes a period of time within which point-in-time recovery must be possible



- Redundancy:** Establishes a fixed number of backups that must be kept



- Retention policies are mutually exclusive.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying a Retention Policy

A *retention policy* describes which backups will be kept and for how long. You can set the value of the retention policy by using the RMAN CONFIGURE command or Enterprise Manager.

Recovery Window Retention Policy

The best practice is to establish a period of time during which it will be possible to discover logical errors and fix the affected objects by doing a point-in-time recovery to just before the error occurred. This period of time is called the *recovery window*. This policy is specified in number of days. For each data file, there must always exist at least one backup that satisfies the following condition:

$$\text{SYSDATE} - \text{backup_checkpoint_time} \geq \text{recovery_window}$$

You can use the following command syntax to configure a recovery window retention policy:

```
RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF <days>
DAYS;
```

where *<days>* is the size of the recovery window.

If you are not using a recovery catalog, then you should keep the recovery window time period less than or equal to the value of the control file parameter `CONTROL_FILE_RECORD_KEEP_TIME` to prevent the record of older backups from being overwritten in the control file. If you are using a recovery catalog, then make sure `CONTROL_FILE_RECORD_KEEP_TIME` is greater than the time period between catalog resynchronizations. Resynchronizations happen when you:

- Create a backup. In this case the synchronization is done implicitly.

Specifying a Retention Policy (continued)

Recovery catalogs are covered in more detail in the lesson titled “Using the RMAN Recovery Catalog.”

Redundancy Retention Policy

If you require a certain number of backups to be retained, then you can set the retention policy on the basis of the redundancy option. This option requires that a specified number of backups be cataloged before any backup is identified as obsolete. The default retention policy has a redundancy of 1, which means that only one backup of a file must exist at any given time. A backup is deemed obsolete when a more recent version of the same file has been backed up.

You can use the following command to reconfigure a redundancy retention policy:

```
RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY <copies>;
```

where <copies> is the number of copies that are required for policy satisfaction.

Disabling the Retention Policy

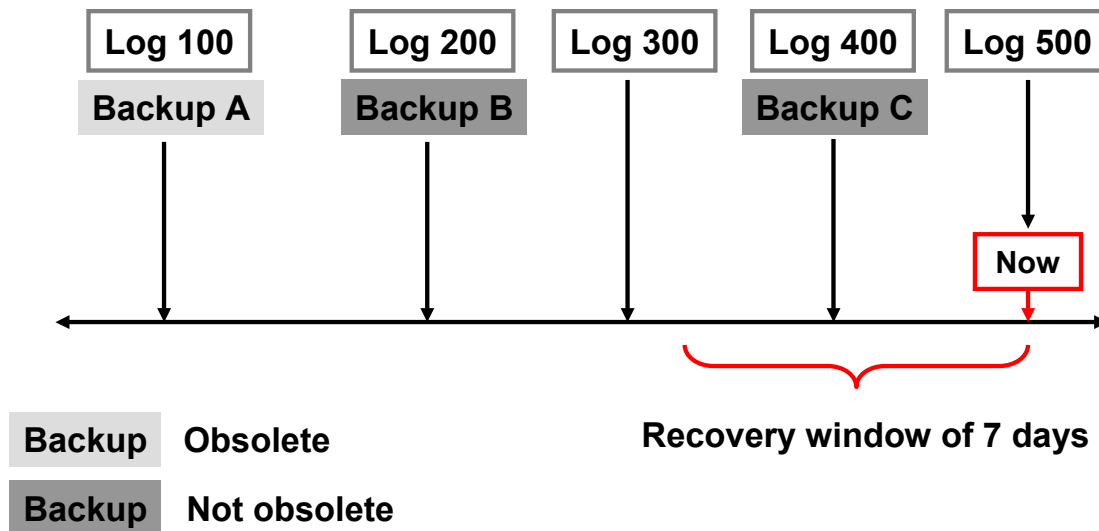
You may want to disable the retention policy totally. If you have a separate system, outside of RMAN, that backs up your disk backups to tape, you may want to do this. If you disable the retention policy, then RMAN never considers a backup obsolete. Because RMAN does not have to decide when to remove a backup from disk (because another utility is managing that), RMAN does not need to be configured for making that decision. In this case, records of each backup are maintained for as long as is specified by the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter.

Disable the retention policy by using this command:

```
RMAN> CONFIGURE RETENTION POLICY TO NONE;
```

Note: You can specify that a backup is an exception to the retention policy that you have defined. This is called an archival backup, which is covered in the lesson titled “Using RMAN to Create Backups.”

A Recovery Window Retention Policy: Example



Backup B and archive logs 201 through 500 are required to satisfy this retention policy.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying a Recovery Window Retention Policy: Example

The retention policy in the slide shows that it requires the ability to recover to any time within the last seven days. Some of the backups and logs are obsolete, because they are not needed to recover to a time within the seven-day window. This retention policy is configured thus:

```
RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
```

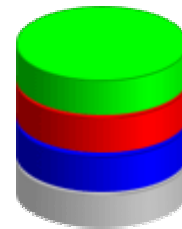
Given the backups and archived log files available, the only data needed to recover to a point inside the recovery window is Backup B and logs 201 through 500. Note that Backup A is not needed because there is a later backup (B) that is still before the recovery window. Also, Backup C is not sufficient as the only backup to retain because it would not satisfy a need to recover to points in time at the beginning of the recovery window. The last backup that was taken before the beginning of the recovery window, including all logs since that backup, is what is necessary.

Using a Fast Recovery Area

- Permanent items:
 - Multiplexed copies of the current control file
 - Multiplexed copies of online redo logs
- Transient items:
 - Archived redo logs
 - Data file copies
 - Control file copies
 - Control file autobackups
 - Backup pieces
 - Flashback logs



Database



Fast Recovery Area

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using a Fast Recovery Area

The Fast Recovery Area is a unified storage location for all recovery-related files and activities in an Oracle database. All files that are needed to completely recover a database from a media failure are part of the Fast Recovery Area. The recovery-related files are of two types: permanent and transient. Permanent files are actively being used by the instance. Transient files are needed only in the event of some type of recovery operation.

Permanent Items

- **Control file:** Depending on the setting of several initialization parameters, a copy of the control file is created in the Fast Recovery Area location when you create a new database or control file. For details see the “Semantics” section of the CREATE CONTROLFILE command in the *Oracle Database SQL Language Reference*.
- **Multiplexed copies of online redo log files:** A mirrored copy from each redo log group can be here. When you create a database, you can specify the location of the online redo log files using the LOGFILE clause. If you do not include that clause, then the locations are set according to the values of the following initialization parameters:
 - **DB_CREATE_ONLINE_LOG_DEST_n:** If one or more of these variables is set, then these are the only locations used.
 - **DB_CREATE_FILE_DEST:** If this is set, then this is the primary file location.
 - **DB_RECOVERY_FILE_DEST:** If this is set, in addition to DB_CREATE_FILE_DEST, then this location is used as the mirror.

Using a Fast Recovery Area (continued)

For more details on how these variables affect the location of the online redo logs, see the LOGFILE clause of the CREATE DATABASE statement in the *Oracle Database SQL Language Reference*.

Transient Items

- **Archived redo log files:** When the Fast Recovery Area is configured, LOG_ARCHIVE_DEST_1 is automatically set to the Fast Recovery Area location. The Archiver background process creates archived redo log files in the Fast Recovery Area and in other configured LOG_ARCHIVE_DEST_n locations. If no LOG_ARCHIVE_DEST_n locations are defined, the default location for archived redo log files is in the Fast Recovery Area.
- **Flashback logs:** Flashback logs are generated when Flashback Database is enabled.
- **Control file autobackups:** The default location for control file autobackups created by RMAN and autobackups generated by the Oracle database server is the Fast Recovery Area.
- **Data file copies:** The BACKUP AS COPY command creates image data file copies in the Fast Recovery Area.
- **RMAN files:** The Fast Recovery Area is the default location that is used by RMAN for backups and restoration of the archive log content from tape for a recovery operation.

Note: If you need a good performance for your FRA, consider creating it on its own physical disks and controllers.

Defining a Fast Recovery Area

The FRA is defined by setting two, dynamic initialization parameters:

- **DB_RECOVERY_FILE_DEST_SIZE**: Sets the disk limit
- **DB_RECOVERY_FILE_DEST**: Sets the location for the FRA

☒ ARCHIVELOG Mode*

Log Archive Filename Format* %t_%s_%r.dbf

Number	Archived Redo Log Destination	Status	Type
1	USE_DB_RECOVERY_FILE_DEST	VALID	Local

Add Another Row

☒ TIP It is recommended that archived redo log files be written to multiple locations spread

Flash Recovery Area Location: +FRA

Flash Recovery Area Size: 6 GB

Flash Recovery Area Size must be set when the location is set.

Non-reclaimable Flash Recovery Area (GB): 1.89

Reclaimable Flash Recovery Area (MB): 609

Free Flash Recovery Area (GB): 3.51

Flash Recovery Area Usage

Component	Size (GB)	Percentage
Backup Piece	1.34	22.4%
Archived Redo Log	0.4	6.6%
Online Log	0.15	2.5%
Control File	0	0%
Image Copy	0	0%
Flashback Log	0	0%
Usable	4.11	68.5%

ORACLE

Copyright © 2009, Oracle. All rights reserved.

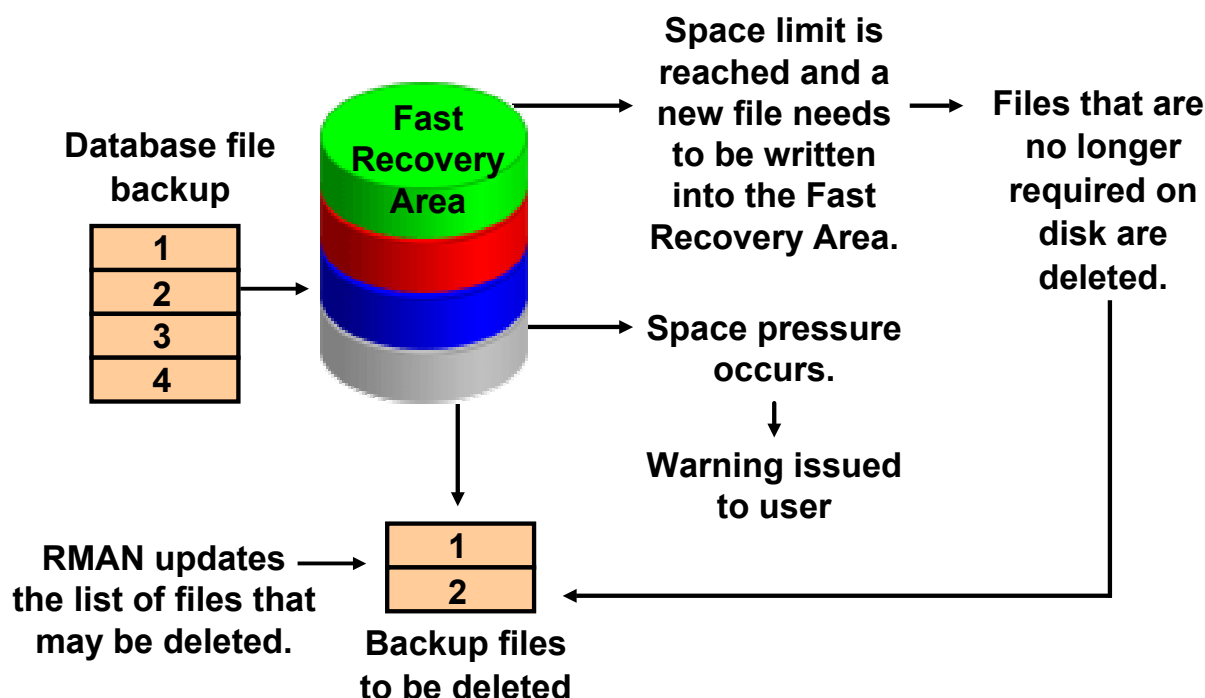
Defining a Fast Recovery Area

Use the following mandatory parameters to define the FRA:

- **DB_RECOVERY_FILE_DEST_SIZE**: You must define a disk limit, which is the amount of space that the FRA is permitted to use. Setting a limit allows the remaining disk space not dedicated to the FRA to be used for other purposes.
 - A basic recommendation for the size of the disk limit is the sum of the database size, the size of incremental backups, and the size of all archive log files that have not been copied to tape. The reason for this recommendation is that the Oracle-suggested backup strategy takes one image copy of the database (without temp files) and then the incremental backups.
 - The minimum size of the FRA should be at least large enough to contain archived redo log files that have not been copied to tape.
 - The sizing of the FRA is dependent on the backup strategy and other options that are implemented. Guaranteed Restore Points also have an effect on the size of the FRA.
- **DB_RECOVERY_FILE_DEST**: An FRA specification contains a location, which is a valid destination to create files.

You can use Enterprise Manager Grid Control and Database Control to easily define the FRA. Navigate to Availability > Recovery Settings. You can define the FRA location and its size on the Recovery Settings page. You must set the size of the FRA when specifying its location.

Fast Recovery Area Space Management



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Fast Recovery Area Space Management

Each time RMAN creates a file in the Fast Recovery Area, the list of files that are no longer required on disk is updated. Based on the value of `DB_RECOVERY_FILE_DEST_SIZE`, when the Fast Recovery Area experiences space pressure or is low on free space because there are no files that can be deleted from the Fast Recovery Area, you are warned of the danger of running out of space. The Oracle database server and RMAN continue to create files in the Fast Recovery Area until 100% of the disk limit is reached. When setting `DB_RECOVERY_FILE_DEST_SIZE`, you must allocate enough space to hold the recovery files, including backups that are waiting to be backed up to tape. Files that are obsolete or have been backed up to tape are likely candidates for deletion to provide free space.

When a file is written into the Fast Recovery Area and space is needed for that file, the Oracle database server deletes a file that is on the obsolete files list. When a file is written and deleted from the Fast Recovery Area, notification is written into the alert log.

Note: When the Fast Recovery Area's used space is at 85%, a warning alert is issued, and when used space is at 97%, a critical alert is issued. These are internal settings and cannot be changed.

A sample alert log output follows:

```
WARNING: db_recovery_file_dest_size of 52428800 bytes is 100.00%
used, and has 0 remaining bytes available.
```

Fast Recovery Area Space Management (continued)

You can issue the following query to determine the action to take:

```
SQL> SELECT object_type, message_type, message_level,  
2 reason, suggested_action  
3 FROM dba_outstanding_alerts;
```

Your choice is to add additional disk space, back up files to a tertiary device, delete files from the Fast Recovery Area using RMAN, or consider changing the RMAN retention policy.

Fast Recovery Area Space Usage

- Configure the retention policy to the minimum value appropriate for your database.
- Back up the archive log files regularly and delete the files upon completion of the backup.
- Optionally, configure an archive redo log deletion policy.
- Use the RMAN REPORT OBSOLETE and DELETE OBSOLETE commands to remove backups and file copies that are not required.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Fast Recovery Area Space Usage

To avoid running out of space in the Fast Recovery Area, perform the following steps as needed or appropriate:

- Use RMAN to delete unnecessary files from the Fast Recovery Area.
- Use RMAN to take frequent backups of the Fast Recovery Area.
- Change the RMAN retention policy to retain backups for a smaller period of time.
- Change the RMAN archived log deletion policy.
- Add disk space and increase the value of the DB_RECOVERY_FILE_DEST_SIZE database initialization parameter if you frequently run out of space.

Enterprise Manager does not report the amount of space used by the Fast Recovery Area on the disk or the amount of space used in the Fast Recovery Area directory tree, but it does report the sizes of the files that RMAN believes are in the directory. So do not put any files in this area that are not managed by RMAN.

If you remove any file from this area with a tool other than RMAN, use RMAN to remove the file entries from the catalog. For example, to back up the archived log files in the Fast Recovery Area and then delete the files after they have been successfully backed up, you would use the RMAN command as follows:

```
BACKUP ARCHIVELOG ALL DELETE ALL INPUT;
```

Fast Recovery Area Space Usage (continued)

If you use a backup solution other than RMAN, you still have to use RMAN to remove the files from the Fast Recovery Area. After the archived redo log files have been backed up and removed from disk, use the RMAN `CROSSCHECK` and `DELETE` commands to reclaim the archived log space from the Fast Recovery Area. You should do this on a regular basis or after every backup.

You can also use the Manage Backups page of Enterprise Manager to manage backups. On that page, you can perform a cross-check operation and delete expired and obsolete backups.

Configuring an Archived Redo Log Deletion Policy

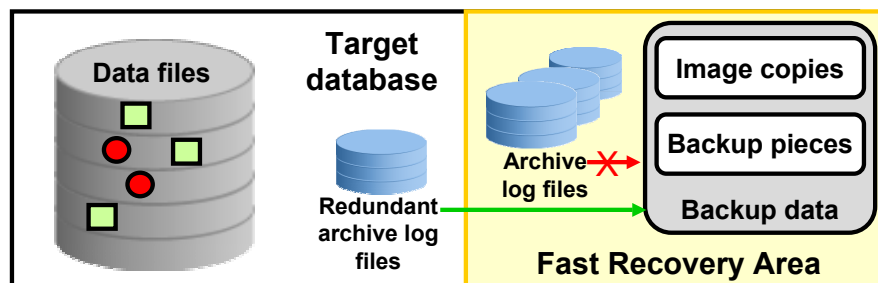
You can use the `CONFIGURE ARCHIVELOG DELETION POLICY` command to specify when archived redo logs are eligible for deletion. This deletion policy applies to *all* archiving destinations, including the FRA.

Archived redo logs can be deleted automatically by the database or as a result of user-initiated RMAN commands:

- Only logs in the FRA can be deleted automatically by the database. For archived redo log files in the FRA, the database retains them as long as possible and automatically deletes eligible logs when additional disk space is required.
- You can manually delete eligible logs from any location, whether inside or outside the FRA, when you issue `BACKUP . . . DELETE INPUT` or `DELETE ARCHIVELOG`.
- The default is:
`CONFIGURE ARCHIVELOG DELETION POLICY TO NONE;`

What Is Done Automatically for You

- Simplified archive log management in a multiple-component environment
- Increased availability by failover of backup to optional destinations



ORACLE

Copyright © 2009, Oracle. All rights reserved.

What Is Done Automatically for You

Simplified Archive Log Management in a Multiple-Component Environment

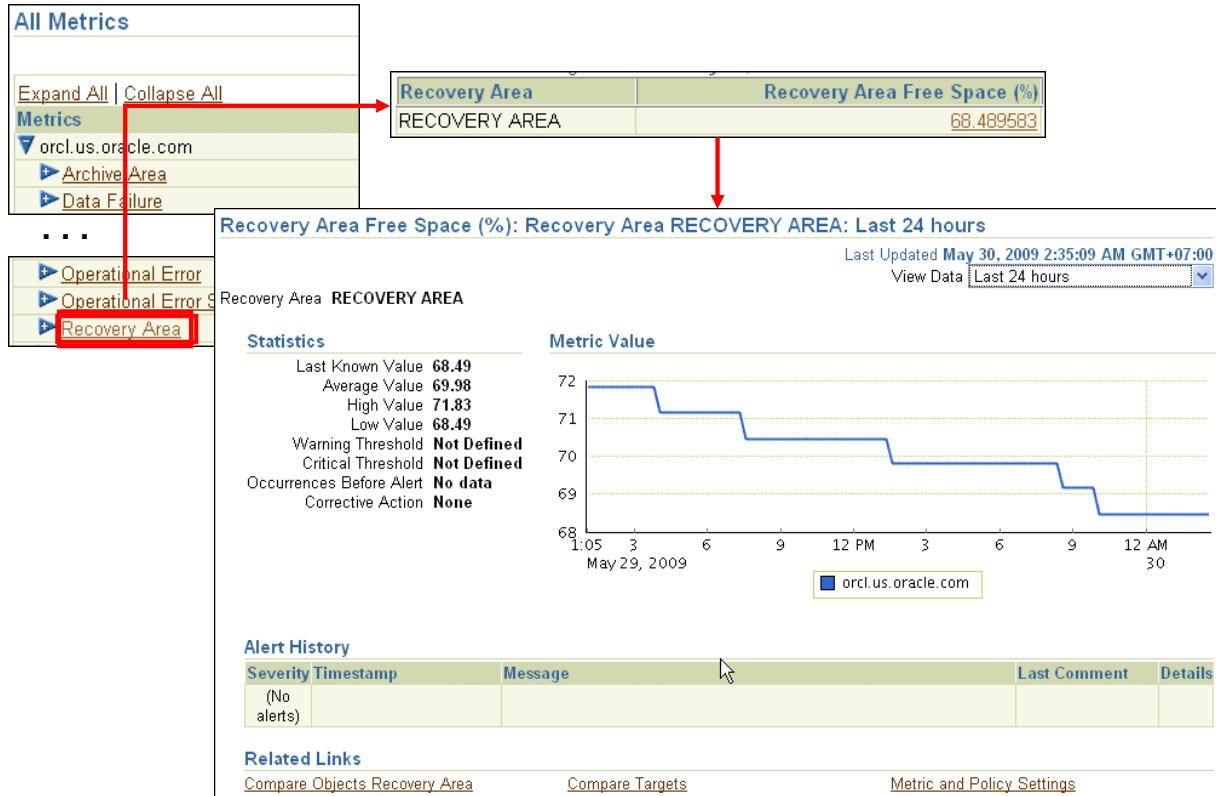
This feature simplifies archive log management when used by multiple components, such as Streams. It also increases availability when backing up archive logs, when an archive log in the FRA is missing or inaccessible.

Enhanced Configuration of Deletion Policies

Archived redo logs are eligible for deletion only when not needed by any required components such as Data Guard, Streams, Flashback Database, and so on. When you configure an archived log deletion policy, the configuration applies to all archiving destinations, including the FRA. Both `BACKUP ... DELETE INPUT` and `DELETE... ARCHIVELOG` use this configuration, as does the FRA.

When you back up the recovery area, RMAN can fail over to other archived redo log destinations if the archived redo log in the FRA is inaccessible or corrupted.

Monitoring the FRA



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring the FRA

Real-time FRA metrics can be viewed through Enterprise Manager Database Control. On the Home page, scroll down to the Related Links section and select All Metrics. Scan the list and click Recovery Area.

The displayed page shows the Recovery Area Free Space (%) metric, which indicates the percentage of the recovery that is free space. Click the percentage number to see the graph of recovery area usage.

Benefits of Using a Fast Recovery Area

Using the Fast Recovery Area for recovery-related files:

- Simplifies the location of database backups
- Automatically manages the disk space allocated for recovery files

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Benefits of Using a Fast Recovery Area

Using a FRA for all recovery-related files simplifies the ongoing administration of your database. Oracle Corporation recommends the use of the Fast Recovery Area for all recovery-related files.

Quiz

Backup sets can be created for:

1. Data files
2. Archive logs
3. Online logs
4. Backup sets

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 2, 4

Quiz

Select all statements that are true about the Fast Recovery Area (FRA):

1. The FRA can use an ASM disk group.
2. The FRA can use an OS directory.
3. The FRA can be used by only one database.
4. The FRA should be put on your slowest disk to improve recovery.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 2

Summary

In this lesson, you should have learned how to:

- Invoke and configure RMAN
- Configure your database in `ARCHIVELOG` mode
- Configure multiple archive log file destinations to increase availability
- Configure the Fast Recovery Area
- Specify a retention policy
- Describe the benefits of using the Fast Recovery Area

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 2 Overview: Configuring for Recoverability

This practice covers the following topics:

- Placing the database in ARCHIVELOG mode
- Verifying that the FRA is configured
- Using RMAN to connect to the target database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

3

Using the RMAN Recovery Catalog

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Contrast the usage of a recovery catalog with that of the control file for the RMAN repository
- Create and configure a recovery catalog
- Register a database in the recovery catalog
- Synchronize the recovery catalog
- Use RMAN stored scripts
- Back up the recovery catalog
- Create a virtual private catalog

ORACLE

Copyright © 2009, Oracle. All rights reserved.

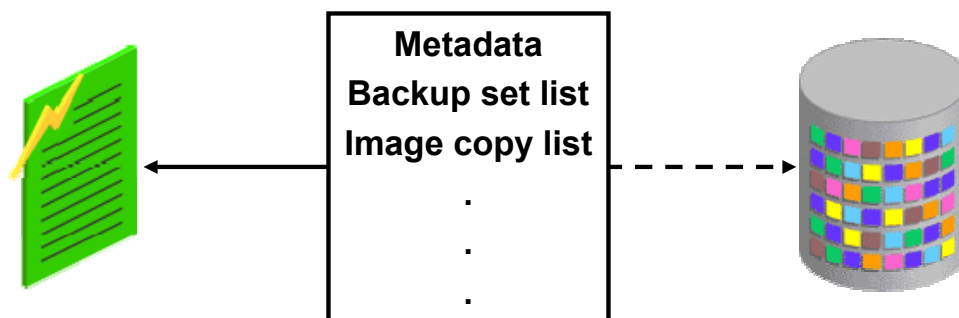
RMAN Repository Data Storage: Comparison of Options

Control file:

- Simpler administration
- Default

Recovery catalog:

- Replicates control file data
- Stores longer history of backups
- Services many targets
- Stores RMAN scripts



ORACLE

Copyright © 2009, Oracle. All rights reserved.

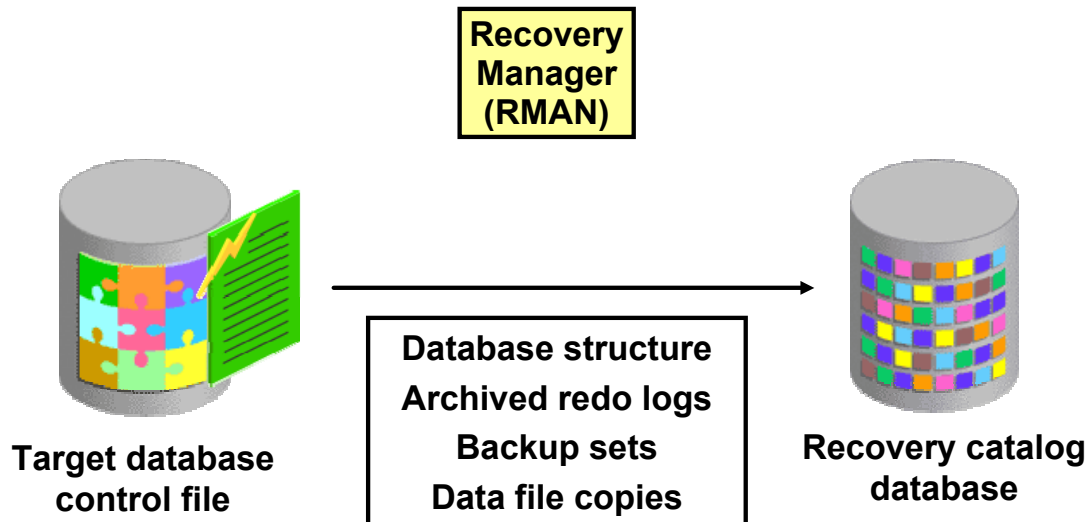
RMAN Repository Data Storage: Comparison of Options

The RMAN repository data is always stored in the control file of the target database. But it can also, additionally, be stored in a separate database called a recovery catalog.

A recovery catalog preserves backup information in a separate database, which is useful in the event of a lost control file. This allows you to store a longer history of backups than what is possible with a control file-based repository. A single recovery catalog is able to store information for multiple target databases. The recovery catalog can also hold RMAN stored scripts, which are sequences of RMAN commands.

If you have very simple backup management requirements, Oracle recommends that you use the control file option rather than a recovery catalog. Having a recovery catalog means you need to manage and back up another database. Therefore, use a recovery catalog only if you can take advantage of the benefits it offers, such as longer backup retention.

Storing Information in the Recovery Catalog



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Storing Information in the Recovery Catalog

RMAN propagates information about the database structure, archived redo logs, backup sets, and data file copies into the recovery catalog from the target database's control file after any operation that updates the repository, and also before certain operations.

Reasons to Use a Recovery Catalog

- Stores more historical information than the control file
- Enables you to use RMAN stored scripts
- Enables you to create customized reports for all registered targets
- Enables you to use the `KEEP FOREVER` clause of the `BACKUP` command
- Allows you to list the data files and tablespaces that are or were in the target database *at a given time*



Copyright © 2009, Oracle. All rights reserved.

Reasons to Use a Recovery Catalog

Although you can use the control file as the sole repository for RMAN, it has finite space for records of backup activities. When you use a recovery catalog, you can store a much longer history of backups. This enables you to perform a recovery that goes back further in time than the history in the control file.

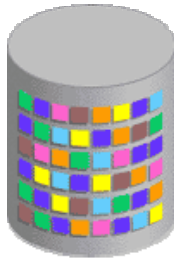
If you want to use RMAN stored scripts, you must use a recovery catalog.

When you use a recovery catalog, the backup and recovery information for all registered targets is contained in one place allowing you to create customized reports by connecting as the recovery catalog owner and querying the various `RC_` views. If you do not use a recovery catalog, you must connect to each target database instance separately and query the `V$` views for the RMAN information in the target control file.

You can use the `BACKUP . . . KEEP` command to create a backup that is retained for a different period of time from that specified by the configured retention policy. The `KEEP FOREVER` clause specifies that the backup or copy never expires and requires the use of a recovery catalog so that the backup records can be maintained indefinitely.

The `REPORT SCHEMA` command lists the tablespaces and data files in the target database. If you add the option of `AT [time|scn|logseq]`, you can see that information at some time in the past. You can use the `AT` option only if you are using a recovery catalog.

Creating the Recovery Catalog: Three Steps



**Configure the
recovery catalog
database.**



**Create the
recovery catalog
owner.**



**Create the
recovery catalog.**

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating the Recovery Catalog: Three Steps

To create a recovery catalog, perform the following three steps:

1. Configure the database in which you want to store the recovery catalog.
2. Create the recovery catalog owner.
3. Create the recovery catalog.

Configuring the Recovery Catalog Database

- Allocate space for the recovery catalog. Consider:
 - Number of databases supported by the recovery catalog
 - Number of archived redo log files and backups recorded
 - Use of RMAN stored scripts
- Create a tablespace for the recovery catalog, which becomes the default tablespace for the recovery catalog owner.

```
SQL> CREATE TABLESPACE rcat_ts DATAFILE SIZE 15M;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring the Recovery Catalog Database

Determine the database in which you will install the recovery catalog schema. Be sure to consider your backup and recovery procedures for this database. Backing up the recovery catalog is covered later in this lesson.

The amount of space required by the recovery catalog schema depends on the number of databases monitored by the catalog. The space increases as the number of archived redo log files and backups for each database increases. If you use RMAN stored scripts, space must be allocated for those scripts. The sample space requirement is 15 MB for each database registered in the recovery catalog.

Creating the Recovery Catalog Owner



- Create the recovery catalog owner.
- Grant the RECOVERY_CATALOG_OWNER role.



```
SQL> CREATE USER rcowner IDENTIFIED BY rcpass
2 TEMPORARY TABLESPACE temp
3 DEFAULT TABLESPACE rcat_ts
4 QUOTA UNLIMITED ON rcat_ts;
SQL> GRANT recovery_catalog_owner TO rcowner;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating the Recovery Catalog Owner

Create a user to serve as the recovery catalog owner. Set the default tablespace for this user to the tablespace you created for the recovery catalog. Be sure to provide UNLIMITED quota on this tablespace for the user. After you have created the user, grant the RECOVERY_CATALOG_OWNER role to the user. The RECOVERY_CATALOG_OWNER role provides privileges for the owner of the recovery catalog. The role includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE PROCEDURE, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE TRIGGER, CREATE TYPE, and CREATE VIEW.

You can use SQL commands or Enterprise Manager to create the user and grant the role.

Creating the Recovery Catalog

- Connect to the recovery catalog database as the catalog owner:

```
$ rman  
RMAN> CONNECT CATALOG username/password@net_service_name
```

- Execute the CREATE CATALOG command:

```
RMAN> CREATE CATALOG;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating the Recovery Catalog

After creating the catalog owner, use the RMAN CREATE CATALOG command to create the catalog tables in the default tablespace of the catalog owner.

Note: As with any database, if the ORACLE_SID environment variable is set to the SID for the recovery catalog database, then there is no need to supply the net_service_name in the CONNECT statement.

Managing Target Database Records in the Recovery Catalog

- Registering a target database in the recovery catalog
- Cataloging additional backup files
- Unregistering a target database from the recovery catalog

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Managing Target Database Records in the Recovery Catalog

Although most information is automatically propagated from the control file to the recovery catalog, there are a few operations you may need to perform to maintain target database records in the recovery catalog.

Registering a Database in the Recovery Catalog

RMAN performs the following actions:

- Creates rows in the recovery catalog tables for the target database
- Copies data from the target database control file to the recovery catalog tables
- Synchronizes the recovery catalog with the control file

```
$ rman TARGET / CATALOG  
    username/password@net_service_name  
RMAN> REGISTER DATABASE;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Registering a Database in the Recovery Catalog

After creating the recovery catalog, you must register the target databases in the recovery catalog. To register your target database, perform the following steps:

1. Invoke RMAN and connect to the recovery catalog database and to the target database as shown in the following example:

```
% rman TARGET / CATALOG rman/rman@reccatdb
```

2. Ensure that the target database is mounted or open.
3. Issue the REGISTER command to register the target database in the recovery catalog:

```
RMAN> REGISTER DATABASE;
```

Using Enterprise Manager to Register a Database

To register a database with a recovery catalog, perform the following steps in Enterprise Manager (EM):

1. Run EM against the target database and navigate to the Recovery Catalog Settings page.
2. Add the recovery catalog to the configuration, if not already listed there.
3. Specify that the target database is to use the recovery catalog chosen in the list.



The EM method of registration also causes EM to use the recovery catalog for backup and recovery–related operations.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Enterprise Manager to Register a Database

To register a database with a recovery catalog in EM, you must first add the recovery catalog to the EM configuration. Running EM on the target database, you select that recovery catalog to be the recovery catalog for the target database.

If you use RMAN to register the database, and do not perform the steps in the slide, then any backup and recovery operations performed using EM will not use the recovery catalog. So, if you plan to use EM, perform the registration steps described here even if you have executed the RMAN REGISTER DATABASE command.

In Enterprise Manager:

1. From the EM Database home page, navigate to Availability > Recovery Catalog Settings. Click Add Recovery Catalog to specify the host, port, and SID of a database with an existing recovery catalog.
2. After you have defined the recovery catalog database, select “Use Recovery Catalog” on the Recovery Catalog Setting page to register the database in the recovery catalog database. When you click OK, the database is registered with the catalog.

Unregistering a Target Database from the Recovery Catalog

- This removes information about the target database from the recovery catalog.
- Use this when you no longer want the target database to be defined in the recovery catalog.

```
$ rman TARGET / CATALOG  
    username/password@net_service_name  
RMAN> UNREGISTER DATABASE;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Unregistering a Target Database from the Recovery Catalog

When you unregister a database from the recovery catalog, all RMAN repository records in the recovery catalog are lost. You can reregister the database. The recovery catalog records for that database are then based on the contents of the control file at the time of re-registration.

Typically, you would unregister a target database only if you no longer want to use the recovery catalog for that database or the database no longer exists.

Note: If you have used Enterprise Manager Database Control to register your database, you must use it again to unregister your database.

Cataloging Additional Backup Files

- CATALOG can be used to catalog existing backup files that are no longer listed in the control file.
- This enables RMAN to use the files during a restore operation.
- Use the CATALOG command to add the following types of backup files to the recovery catalog:
 - Control file copies
 - Data file copies
 - Backup pieces
 - Archived redo log files

```
RMAN> CATALOG BACKUPPIECE 'file_name';
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Cataloging Additional Backup Files

If you have additional control file copies, data file copies, backup pieces, or archived redo log files on disk, you can catalog them in the recovery catalog by using the CATALOG command. If backups have aged out of the control file, you can catalog them so that RMAN can use them during a restore operation.

Examples of cataloging a control file, data file, archived redo log file, and backup piece follow:

```
RMAN> CATALOG CONTROLFILECOPY
'/disk1/controlfile_bkup/2009_01_01/control01.ctl';
RMAN> CATALOG DATAFILECOPY
'/disk1/datafile_bkup/2009_01_01/users01.dbf';
RMAN> CATALOG ARCHIVELOG '/disk1/arch_logs/archive1_731.log',
'/disk1/arch_logs/archive1_732.log';
RMAN> CATALOG BACKUPPIECE '/disk1/backups/backup_820.bkp';
```

You can catalog all files in the currently enabled Flash Recovery Area as follows:

```
RMAN> CATALOG RECOVERY AREA NOPROMPT;
```

START WITH Option

Use the START WITH option to catalog all files found in the directory tree specified. Provide a prefix that indicates the directory and possibly a file prefix to look for. You cannot use wildcards; this is only a prefix.

Cataloging Additional Backup Files (continued)

All types of backup files that are found in the specified directory and subdirectories are cataloged. Suppose you have several backup files in the `/tmp/arch_logs` directory. The following command catalogs all of them:

```
RMAN> CATALOG START WITH '/tmp/arch_logs/';
```

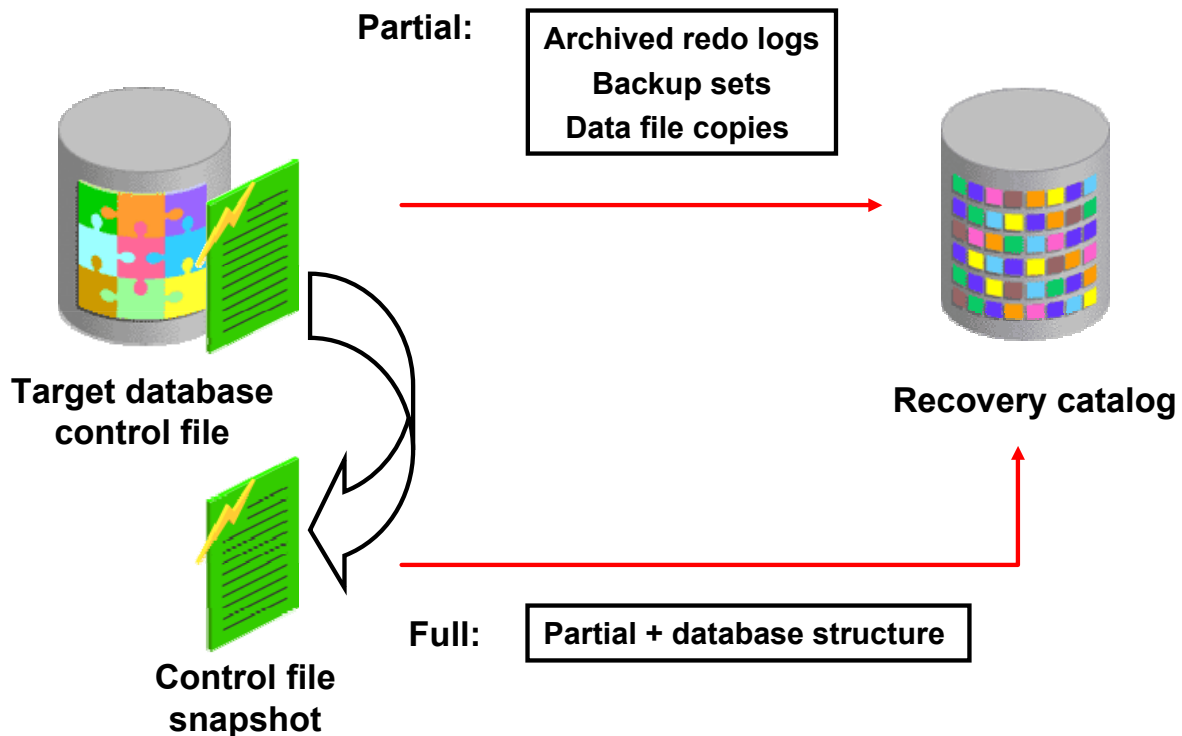
Suppose you want to be sure to catalog only those files in the `/tmp` directory whose file names start with the string `bset`. The following accomplishes that:

```
RMAN> CATALOG START WITH '/tmp/bset';
```

This command also catalogs any backup files that are found in directory trees that begin with `/tmp/bset`.

The `CATALOG` command can be used without being connected to a recovery catalog.

Recovery Catalog Resynchronization: Concepts



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovery Catalog Resynchronization: Concepts

When RMAN performs a *resynchronization*, it compares the recovery catalog to either the current control file of the target database or a backup/standby control file and updates the recovery catalog with information that is missing or changed.

There are two types of resynchronization: partial and full. For partial resynchronization, RMAN compares the control file to the recovery catalog and updates the recovery catalog with any metadata concerning backups, archived redo logs, data file copies, and so on. For a full synchronization, RMAN first creates a control file snapshot, which is simply a temporary copy of the control file. It uses the snapshot to make the comparison to the recovery catalog. It compares and updates all the data that a partial resynchronization does, but it also includes any database structure changes. For example, database schema changes or new tablespaces are included in a full resynchronization.

Note: The database schema includes names and locations of data files, redo log files, archive log files, undo segments, and other information found in the control file.

If the only changes to the control file are those records that are governed by `CONTROL_FILE_RECORD_KEEP_TIME`, then a partial resynchronization is done. Otherwise, a full resynchronization is done. A full resynchronization is also done when you issue the `RESYNC CATALOG` command, which is described in the next slide.

Manually Resynchronizing the Recovery Catalog

Manually resynchronize the recovery catalog in the following situations:

- After the recovery catalog was unavailable for RMAN to automatically resynchronize it
- When you perform infrequent backups of your target database
- After making changes to the physical structure of the target database

```
RMAN> RESYNC CATALOG;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Manually Resynchronizing the Recovery Catalog

Perform a manual resynchronization of the recovery catalog in the following situations:

- If the recovery catalog was unavailable when you issued RMAN commands that cause a partial resynchronization
- If you perform infrequent backups of your target database because the recovery catalog is not updated automatically when a redo log switch occurs or when a redo log is archived
- After making any change to the physical structure of the target database

Note: Refer to the *Backup and Recovery User's Guide* for detailed information about records that are updated during resynchronization.

Using RMAN Stored Scripts

Stored scripts are:

- An alternative to command files
- Available to any RMAN client that can connect to the target database and recovery catalog
- Of two types:
 - Local: Associated with the

```
CREATE SCRIPT script_name  
{ <RMAN commands> }
```

 target database to which RMAN is connected when the script is created
 - Global: Can be executed

```
CREATE GLOBAL SCRIPT script_name  
{ <RMAN commands> }
```

 against any database registered in the recovery catalog
- Created from a text file (additional option)

```
CREATE [GLOBAL] SCRIPT script_name FROM FILE 'file_name';
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using RMAN Stored Scripts

You can use RMAN stored scripts as an alternative to command files for managing frequently used sequences of RMAN commands. Unlike command files that are available only on the system on which they are stored, a stored script is always available to any RMAN client that can connect to the target database and recovery catalog.

Stored scripts can be defined as global or local. A local stored script is associated with the target database to which RMAN is connected when the script is created, and can be executed only when you are connected to that target database. A global stored script can be executed against any database registered in the recovery catalog, if the RMAN client is connected to the recovery catalog and a target database.

Creating RMAN Stored Scripts

Connect to the desired target database and the recovery catalog and execute the CREATE SCRIPT command to create a stored script.

Executing RMAN Stored Scripts

- Executing a script:

```
RUN { EXECUTE SCRIPT
      script_name
    ; }
```

- Executing a global script:

```
RUN { EXECUTE GLOBAL SCRIPT
      script_name
    ; }
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Executing RMAN Stored Scripts

Connect to the target database and recovery catalog, and use the EXECUTE SCRIPT command to execute a stored script. Note that the EXECUTE SCRIPT command requires a RUN block. If an RMAN command in the script fails, subsequent RMAN commands in the script do not execute. When you execute the script, it uses the automatic channels configured at the time. Use ALLOCATE CHANNEL commands in the script if you need to override the configured channels as shown in the following example:

```
RMAN> RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL ch2 DEVICE TYPE DISK;
  ALLOCATE CHANNEL ch3 DEVICE TYPE DISK;
  EXECUTE SCRIPT full_backup;
}
```

Maintaining RMAN Stored Scripts

- Displaying a script:

```
PRINT [GLOBAL] SCRIPT script_name;
```

- Sending the contents of a script to a file:

```
PRINT [GLOBAL] SCRIPT script_name TO FILE 'file_name';
```

- Displaying the names of defined scripts:

```
LIST [GLOBAL] SCRIPT NAMES;
```

- Displaying a script:

```
REPLACE [GLOBAL] SCRIPT script_name  
{ <RMAN commands> ; }
```

- Updating a script from a file:

```
REPLACE [GLOBAL] SCRIPT script_name FROM FILE  
'file_name';
```

- Deleting a script:

```
DELETE SCRIPT script_name;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Displaying RMAN Stored Script Information

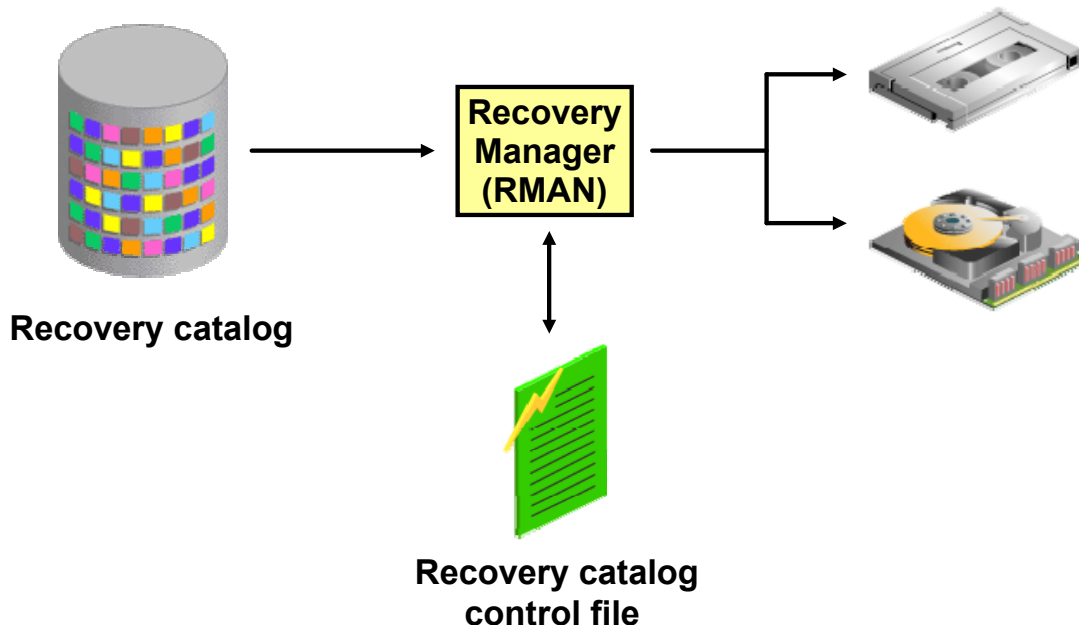
Connect to the target database and recovery catalog and use the PRINT SCRIPT command to display a stored script or write it out to a file.

Use the LIST SCRIPT NAMES command to display the names of scripts defined in the recovery catalog. This command displays the names of all stored scripts, both global and local, that can be executed for the target database to which you are currently connected.

Connect to the target database and recovery catalog and use the REPLACE SCRIPT command to update stored scripts. RMAN creates the script if it does not exist.

To delete a stored script from the recovery catalog, connect to the catalog and a target database, and use the DELETE SCRIPT command.

Backing Up the Recovery Catalog



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Backing Up the Recovery Catalog

The recovery catalog is an Oracle database, so it needs to be backed up as any database should. Oracle recommends using RMAN to back it up and, of course, use the control file as the RMAN repository, instead of a recovery catalog. Never store a recovery catalog containing the RMAN repository for a database in the same database as the target database or on the same disks as the target database. A recovery catalog is effective only if it is separated from the data that it is designed to protect.

Configure control file autobackup so that the control file is backed up every time a backup is made of the recovery catalog. Any time you make a backup in the target database, back up the recovery catalog right afterward. This protects the record of the latest backup.

Below is a summary of how to configure the backup and recovery environment for your recovery catalog:

- Run the recovery catalog in ARCHIVELOG mode.
- Set the retention policy to a REDUNDANCY value greater than one.
- Back up the recovery catalog to disk and tape.
- To make the backups, use the `BACKUP DATABASE PLUS ARCHIVELOG` command.
- Use the control file (NOCATALOG), not another recovery catalog, as the RMAN repository.
- Configure control file autobackup to be ON.

Re-Creating an Unrecoverable Recovery Catalog

To partially re-create the contents of a lost recovery catalog, use the following commands:

- **RESYNC CATALOG** command: Updates the recovery catalog with any RMAN repository information from the control file of the target database or a control file copy
- **CATALOG START WITH** command: Recatalogs any available backups

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Re-Creating an Unrecoverable Recovery Catalog

If the recovery catalog database is lost or damaged, and recovery of the recovery catalog database through the normal Oracle recovery procedures is not possible, then you must re-create the catalog.

You can use the following commands to partially repopulate the contents of the recovery catalog:

- **RESYNC CATALOG:** Use this command to update the recovery catalog with any RMAN repository information from the control file of the target database or a control file copy. Note that any metadata from control file records that aged out of the control file is lost.
- **CATALOG START WITH . . . :** Use this command to recatalog any available backups.

Exporting and Importing the Recovery Catalog

Use the Export and Import utilities or the Data Pump utilities to:

- Move the recovery catalog from one database to another
- Create a logical backup of the RMAN recovery catalog

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Exporting and Importing the Recovery Catalog

You can use Export and Import to move the recovery catalog from one database to another.

You can also create an export of the recovery catalog to serve as a logical backup.

Perform the following steps to export a recovery catalog from one database and import the catalog into a second database:

1. Use one of the Oracle Export utilities to export the catalog data from the database.
2. Create a recovery catalog user on the database you are exporting to and grant the user necessary privileges.
3. Use the corresponding Import utility to import the catalog data into the schema created in step 2.

You should not execute the `CREATE CATALOG` command before or after importing the catalog into the database. The import operation creates the catalog in the second database.

Note: The recovery catalog can be backed up and moved to another database as a transportable tablespace by using Export and Import or Data Pump, as well as using logical methods.

Upgrading and Dropping the Recovery Catalog

To upgrade the recovery catalog to the version required by the RMAN client, use the `UPGRADE CATALOG` command:

```
UPGRADE CATALOG;
```

To drop the recovery catalog schema, use the `DROP CATALOG` command:

```
DROP CATALOG;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Upgrading and Dropping the Recovery Catalog

If you use a version of the recovery catalog that is older than that required by the RMAN client, then you must upgrade it by executing the `UPGRADE CATALOG` command.

- To install the new recovery catalog schema, the recovery catalog user must have the `CREATE TYPE` privilege.
- You must be connected to the catalog database, and the catalog database must be open. You do not have to be connected to the target database.
- You must enter the `UPGRADE` command a second time to confirm the upgrade. You receive an error if the recovery catalog is already at a version greater than that needed by the RMAN executable. However, RMAN permits the command to be run if the recovery catalog is current, so that the packages can be re-created if necessary.
- RMAN displays all error messages generated during the upgrade in the message log.

If you no longer want to maintain a recovery catalog, you can drop the recovery catalog schema from the tablespace with the `DROP CATALOG` command. Dropping the catalog deletes the recovery catalog record of backups for all target databases registered in the catalog.

- Execute this command only at the RMAN prompt.
- You must be connected to the recovery catalog database through the `CATALOG` command-line option or the `CONNECT CATALOG` command. The catalog database must be open. You do not have to be connected to the target database.
- Enter the command twice to confirm that you want to drop the schema.

IMPORT CATALOG Command

1. Connecting to the destination recovery catalog:

```
CONNECT CATALOG cat111/oracle@destdb;
```

2. Importing metadata for all registered databases:

```
IMPORT CATALOG cat102/oracle@srcdb;
```

3. Importing metadata for two registered databases:

```
IMPORT CATALOG cat92/oracle@catdb DBID=1423241, 1423242;
```

4. Importing metadata from multiple catalogs:

```
IMPORT CATALOG cat102/rman@srcdb;  
IMPORT CATALOG cat101/rman@srcdb;  
IMPORT CATALOG cat92/rman@srcdb NO UNREGISTER;
```

Must be equal to the
current version of the
RMAN executable

ORACLE

Copyright © 2009, Oracle. All rights reserved.

IMPORT CATALOG Command

With the `IMPORT CATALOG` command, you can import the metadata from one recovery catalog schema into a different catalog schema. If you created catalog schemas of different versions to store metadata for multiple target databases, this command enables you to maintain a single catalog schema for all databases.

```
IMPORT CATALOG <connectStringSpec>  
[DBID = <dbid> [, <dbid>, ...]]  
[DB_NAME=<dbname>[, <dbname>, ...]]  
[ NO UNREGISTER ];
```

<connectStringSpec> is the source recovery catalog connect string. The version of the source recovery catalog schema must be equal to the current version of the RMAN executable. If needed, upgrade the source catalog to the current RMAN version.

DBID: You can specify the list of database IDs whose metadata should be imported from the source catalog schema. When not specified, RMAN merges metadata for all database IDs from the source catalog schema into the destination catalog schema. RMAN issues an error if the database whose metadata is merged is already registered in the recovery catalog schema.

The IMPORT CATALOG Command (continued)

DB_NAME: You can specify the list of database names whose metadata should be imported. If the database name is ambiguous, RMAN issues an error.

NO UNREGISTER: By default, the imported database IDs are unregistered from the source recovery catalog schema after a successful import. By using the NO UNREGISTER option, you can force RMAN to keep the imported database IDs in the source catalog schema.

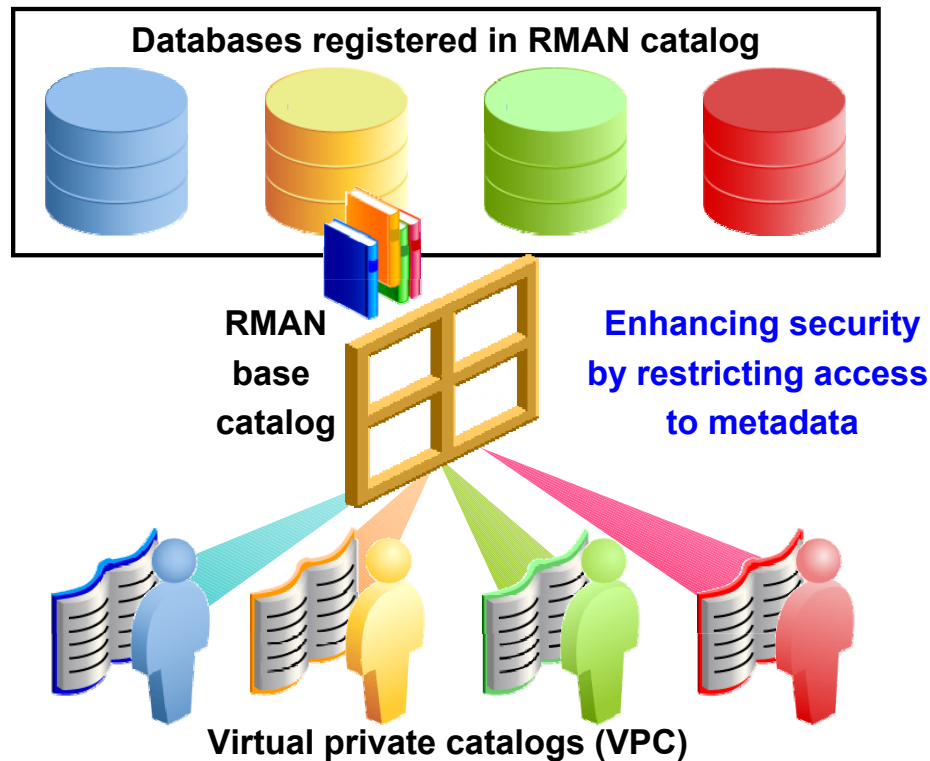
Import Examples (continued)

1. RMAN must be connected to the destination recovery catalog—for example, the `cat111` schema—which is the catalog into which you want to import catalog data. This is the first step in all examples given in the slide.
Note: The version of the source recovery catalog schema must be equal to the current version of the RMAN executable. Before you import catalogs of earlier versions, you must upgrade them to the version of your RMAN executable.
2. In this example, the `cat102` user owns an RMAN catalog in the `srcdb` database. You want RMAN to import all registered databases and unregister them in the source catalog.
3. The `cat92` user owns an RMAN catalog in the `srcdb` database. You want RMAN to import the databases with the DBID 1423241 and 1423242, and unregister them in the source catalog.
4. The `srcdb` database contains three different recovery catalogs. RMAN imports metadata for all database IDs (registered in these catalogs) into the `cat111` schema in the `destdb` database. All imported target databases are unregistered from their source catalogs except for the databases registered in the `cat92` schema.

Additional Usage Details

- Ensure that no target database is registered in both the source catalog schema and the destination catalog schema. If a target database is registered in both schemas, then unregister this database from the source catalog and retry the import.
- If the operation fails in the middle of the import, then the import is rolled back. There is never a state of partial import.
- When stored scripts in the source and destination catalog schemas have name conflicts, RMAN renames the stored script of the source catalog schema.

Creating and Using Virtual Private Catalogs



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating and Using Virtual Private Catalogs

This feature allows a consolidation of RMAN repositories and maintains a separation of responsibilities, which is a basic security requirement.

The RMAN catalog has been enhanced to create virtual private RMAN catalogs for groups of databases and users. The catalog owner creates the base catalog and grants the `RECOVERY_CATALOG_OWNER` privilege to the owner of the virtual catalog. The catalog owner can either grant access to a registered database or grant the `REGISTER` privilege to the virtual catalog owner. The virtual catalog owner can then connect to the catalog for a particular target or register a target database. After this configuration, the VPC owner uses the virtual private catalog just like a standard base catalog.

As the catalog owner, you can access all the registered database information in the catalog. You can list all databases registered with the `SQL*Plus` command:

```
SELECT DISTINCT db_name FROM DBINC;
```

As the virtual catalog owner, you can see only the databases to which you have been granted access.

Note: If a catalog owner has not been granted `SYSDBA` or `SYSOPER` on the target database, most RMAN operations cannot be performed.

Using RMAN Virtual Private Catalogs

1. Create an RMAN base catalog:

```
RMAN> CONNECT CATALOG catowner/oracle@catdb  
RMAN> CREATE CATALOG;
```

2. Grant RECOVERY_CATALOG_OWNER to VPC owner:

```
SQL> CONNECT SYS/oracle@catdb AS SYSDBA  
SQL> GRANT RECOVERY_CATALOG_OWNER to vpcowner;
```

- 3a. Grant REGISTER to the VPC owner:

```
RMAN> CONNECT CATALOG catowner/oracle@catdb  
RMAN> GRANT REGISTER DATABASE TO vpcowner;
```

- 3b. Or, grant CATALOG FOR DATABASE to the VPC owner:

```
RMAN> GRANT CATALOG FOR DATABASE db10g TO vpcowner;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using RMAN Virtual Private Catalogs

You create virtual private RMAN catalogs for groups of databases and users.

1. The catalog owner creates the base catalog.
2. The DBA on the catalog database creates the user that will own the virtual private catalog (VPC) and grants him or her the RECOVERY_CATALOG_OWNER privilege.
3. The base catalog owner can grant access for previously registered databases to the VPC owner or grant REGISTER to the VPC owner. The GRANT CATALOG command is:

```
GRANT CATALOG FOR DATABASE prod1, prod2 TO vpcowner;
```

The GRANT REGISTER command is:

```
GRANT REGISTER DATABASE TO vpcowner;
```

The virtual catalog owner can then connect to the catalog for a particular target or register a target database. After the VPC is configured, the VPC owner uses it just like a standard base catalog.

Using RMAN Virtual Private Catalogs

4a. Create a virtual catalog for 11g clients:

```
RMAN> CONNECT CATALOG vpcowner/oracle@catdb  
RMAN> CREATE VIRTUAL CATALOG;
```

4b. Or, create a virtual catalog for pre-11g clients:

```
SQL> CONNECT vpcowner/oracle@catdb  
SQL> exec catowner.dbms_rcvcat.create_virtual_catalog;
```

5. Register a new database in the catalog:

```
RMAN> CONNECT TARGET / CATALOG vpcowner/oracle@catdb  
RMAN> REGISTER DATABASE;
```

6. Use the virtual catalog:

```
RMAN> CONNECT TARGET / CATALOG vpcowner/oracle@catdb;  
RMAN> BACKUP DATABASE;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using RMAN Virtual Private Catalogs (continued)

4. Create a virtual private catalog.
 - a. If the target database is an Oracle Database 11g database and the RMAN client is an 11g client, you can use the RMAN command:
CREATE VIRTUAL CATALOG;
 - b. If the target database is Oracle Database 10g Release 2 or earlier (using a compatible client), you must execute the supplied procedure from SQL*Plus:
BASE_CATALOG_OWNER.DBMS_RCVCAT.CREATE_VIRTUAL_CATALOG;
5. Connect to the catalog using the VPC owner login, and use it as a normal catalog.
6. The virtual catalog owner can see only those databases that privileges have been granted for. For most RMAN operations, you additionally need the SYSDBA or SYSOPER privileges on the target database.

Recovery Catalogs Summary

Managing recovery catalogs:

1. Create the recovery catalog.
2. Register your target databases in the recovery catalog.
3. If desired, merge recovery catalogs using the new `IMPORT CATALOG` command.
4. If needed, catalog any older backups.
5. If needed, create new virtual recovery catalogs for specific users.
6. Protect the recovery catalog.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovery Catalogs Summary

The basic workflow of managing recovery catalogs is not new. However, it has been enhanced by two important features: the consolidation of RMAN repositories and virtual private catalogs, which allow a separation of responsibilities.

1. Create the recovery catalog.
2. Register your target databases in the recovery catalog. This step enables RMAN to store metadata for the target databases in the recovery catalog.
3. If desired, you can also use the `IMPORT CATALOG` command to merge recovery catalogs.
4. If needed, catalog any older backups, whose records are no longer stored in the target control file.
5. If needed, create virtual recovery catalogs for specific users and determine the metadata to which they are permitted access.
6. Protect the recovery catalog by including it in your backup and recovery strategy.

Recovery Catalogs Summary (continued)

The recovery catalog contains metadata about RMAN operations for each registered target database. The catalog includes the following types of metadata:

- Data file and archived redo log backup sets and backup pieces
- Data file copies
- Archived redo logs and their copies
- Tablespaces and data files on the target database
- Stored scripts, which are named user-created sequences of RMAN commands
- Persistent RMAN configuration settings

The enrolling of a target database in a recovery catalog for RMAN use is called registration. The recommended practice is to register all your target databases in a single recovery catalog. For example, you can register the `prod1`, `prod2`, and `prod3` databases in a single catalog owned by the `catowner` schema in the `catdb` database.

The owner of a centralized recovery catalog, which is also called the base recovery catalog, can grant or revoke restricted access to the catalog to other database users. All metadata is stored in the base catalog schema.

Each restricted user has full read-write access to his or her own metadata, which is called a virtual private catalog.

The recovery catalog obtains crucial RMAN metadata from the control file of each registered target database. The resynchronization of the recovery catalog ensures that the metadata that RMAN obtains from the control files is current.

You can use a stored script as an alternative to a command file for managing frequently used sequences of RMAN commands. The script is stored in the recovery catalog rather than on the file system. A local stored script is associated with the target database to which RMAN is connected when the script is created, and can be executed only when you are connected to this target database. A global stored script can be run against any database registered in the recovery catalog.

You can use a recovery catalog in an environment in which you use or have used different versions of the database. As a result, your environment can have different versions of the RMAN client, recovery catalog database, recovery catalog schema, and target database. **You can merge one recovery catalog (or metadata for specific databases in the catalog) into another recovery catalog for ease of management.**

Quiz

Select all statements that are true about the Oracle recovery catalog:

1. The recovery catalog allows you to store a longer history of backups than what is possible with a control file–based repository.
2. Oracle recommends that you use the recovery catalog for all databases without exception.
3. You must use the EM method of registration in order to use the recovery catalog for backup and recovery–related operations within EM.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 3

Quiz

The RMAN catalog schema may be backed up by using Data Pump Export.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Summary

In this lesson, you should have learned how to:

- Contrast the usage of a recovery catalog with that of the control file for the RMAN repository
- Create and configure a recovery catalog
- Register a database in the recovery catalog
- Synchronize the recovery catalog
- Use RMAN stored scripts
- Back up the recovery catalog
- Create a virtual private catalog

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 3 Overview: Using the RMAN Recovery Catalog

This practice covers the following topics:

- Creating the `RCAT` recovery catalog instance with the `dbca` utility
- Creating the Recovery catalog owner and granting privileges
- Creating a recovery catalog in RMAN
- Registering the `ORCL` database
- Backing up the `RCAT` recovery catalog

ORACLE

Copyright © 2009, Oracle. All rights reserved.

4

Configuring Backup Settings

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use Enterprise Manager to configure backup settings
- Enable control file autobackup
- Configure backup destinations
- Allocate channels for tape destination
- Configure backup optimization
- Create a compressed backup
- Create an encrypted backup

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring Persistent Settings for RMAN

- RMAN is preset with default configuration settings.
- Use the `CONFIGURE` command to:
 - Configure automatic channels
 - Specify the backup retention policy
 - Specify the number of backup copies to be created
 - Set the default backup type to `BACKUPSET` or `COPY`
 - Limit the size of backup pieces
 - Exempt a tablespace from backup
 - Enable and disable backup optimization
 - Configure automatic backups of control files
 - Define the archivelog deletion policy
 - Specify the parallelism for a device
 - Set the encryption and compression parameters to be used for backups

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring Persistent Settings for RMAN

To simplify ongoing use of RMAN for backup and recovery, RMAN enables you to set several persistent configuration settings for each target database. These settings control many aspects of RMAN's behavior. You can save persistent configuration information such as channel parameters, parallelism, and the default device type in the RMAN repository. These configuration settings are always stored in the control file and in the recovery catalog database (if it exists).

These settings have default values, which allow you to use RMAN immediately. However, as you develop a more advanced backup and recovery strategy, you may have to change these settings to implement that strategy. You can use the `CONFIGURE` command to configure persistent settings for RMAN backup, restore, duplication, and maintenance jobs. These settings are in effect for any RMAN session until the configuration is cleared or changed.

Note: The configuration settings can be changed in an RMAN job (or session) just for the duration of the job (or session) with the `SET` command.

EM Note: The same is true for using RMAN via the Enterprise Manager interface. The backup settings provide the default settings for all backups taken. When creating a backup, some of these settings can be overridden for that specific backup.

Viewing Persistent Settings

To examine the persistent RMAN settings for a database:

- Connected only to the target, you enter `SHOW ALL` at the RMAN prompt.

Or:

- Logged in to the target database instance, you query the `V$RMAN_CONFIGURATION` view.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Viewing Persistent Settings

You can view RMAN persistent settings, when you are connected to the target and enter the `SHOW ALL` command, or when you are logged in to SQL*Plus and query the `V$RMAN_CONFIGURATION` view.

Example:

```
SQL> select * from V$RMAN_CONFIGURATION
2 /
```

CONF#	NAME	VALUE
1	CONTROLFILE AUTOBACKUP	ON
2	CHANNEL	DEVICE TYPE 'SBT_TAPE' PARMS
	'SBT_LIBRARY=oracle.disksbt, ENV= (BACKUP_DIR=/tape)	

Control File Autobackups

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

Backup Settings	
Device	Backup Set
Policy	
Backup Policy	
<input checked="" type="checkbox"/> Automatically backup the control file and server parameter file (SPFILE) with every backup and database structural change	
Autobackup Disk Location	<div>An existing directory or diskgroup name where the control file and server parameter file will be backed up. If you do not specify a location, the files will be backed up to the flash recovery area location.</div>

Best Practice Tip: Oracle recommends that you enable control file autobackup.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Control File Autobackups

You can use Oracle Enterprise Manager to specify the backup settings for an instance. From the Database Home page, navigate to Availability > Backup Settings.

To easily recover from the loss of all control file copies, you should configure RMAN to take automatic backups of the control file. The automatic backup of the control file occurs independently of any backup of the current control file explicitly requested as part of your backup command. If you are running RMAN in NOCATALOG mode, it is highly recommended that you activate control file autobackup. Otherwise, if you lose your control file, your database may be unrecoverable.

To configure control file autobackup, modify the backup policy for your database by using Enterprise Manager or use the following RMAN command:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

By default, control file autobackups are disabled. If you enable control file autobackups, then RMAN automatically backs up the control file and the current server parameter file (if used to start up the database) under the following circumstances:

- At the end of a run script
- When a successful backup is recorded in the RMAN repository
- When a structural change of the database occurs the Oracle kernel itself makes the backup (for example, after DDL operations that affect the content of the control file)

Control File Autobackups (continued)

The control file autobackup file name has a default format of %F for all device types, so that RMAN can infer the file location and restore it without a repository. This variable format translates into c-
IIIIIIIIII-YYYYMMDD-QQ, where:

- IIIIIIIIIII stands for the DBID
- YYYYMMDD is a time stamp of the day the backup is generated
- QQ is the hex sequence that starts with 00 and has a maximum of FF

You can change the default format by using the `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE type TO 'string'` command. The value of string must contain the substitution variable %F and cannot contain other substitution variables. For example:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT
FOR DEVICE TYPE DISK TO '/u01/oradata/cf_ORCL_auto_%F';
```

Control file autobackups are stored in the Flash Recovery Area, unless otherwise specified.

With a control file autobackup, RMAN can recover the database even if the current control file, recovery catalog, and server parameter file are inaccessible. Because the path used to store the autobackup follows a well-known format, RMAN can search for and restore the server parameter file or control file from that autobackup.

Managing Persistent Settings

- Use multiple streams of data to and from a device:

```
RMAN> CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
```

- Use the SHOW command to list current settings:

```
RMAN> SHOW CONTROLFILE AUTOBACKUP FORMAT;  
RMAN> SHOW EXCLUDE;  
RMAN> SHOW ALL;
```

- Use the CLEAR option of the CONFIGURE command to reset any persistent setting to its default value:

```
RMAN> CONFIGURE BACKUP OPTIMIZATION CLEAR;  
RMAN> CONFIGURE MAXSETSIZE CLEAR;  
RMAN> CONFIGURE DEFAULT DEVICE TYPE CLEAR;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Managing Persistent Settings

Parallelism is the number of streams of data that can be used to read from and write to the device. This effectively causes that number of channels to be allocated when the device is used by RMAN. For example, if a media manager has two tape drives available, then parallelism 2 would allow both tape drives to be used simultaneously for BACKUP commands using that media manager. Parallelism for the disk device type is also useful, when you want to spread out a backup over multiple disks.

Specify the parallelism to be used on the device using the PARALLELISM clause, like this:

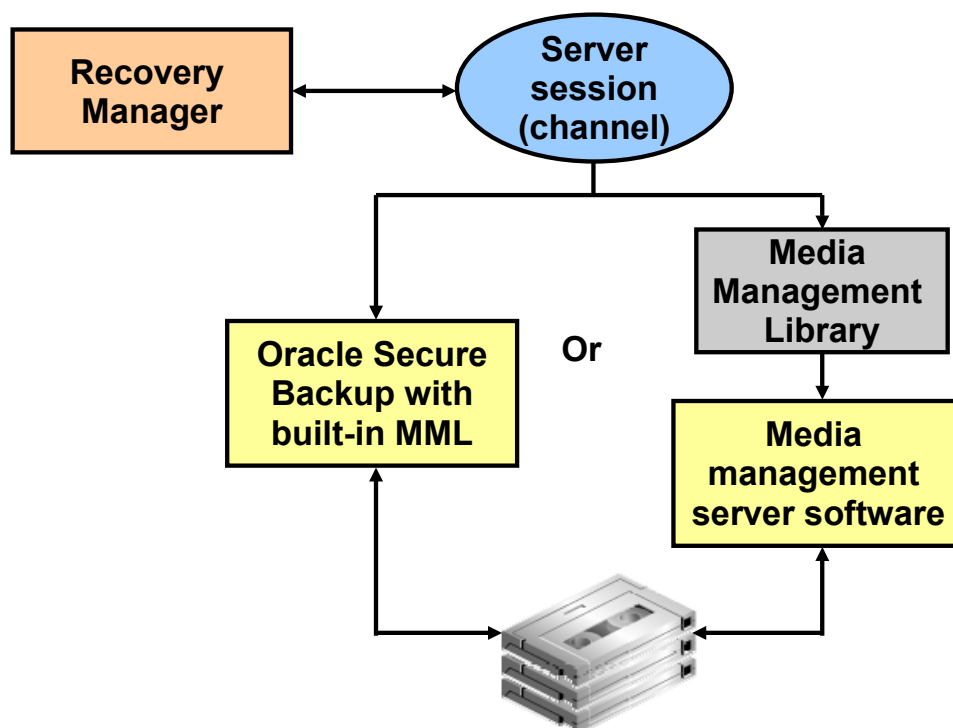
```
CONFIGURE DEVICE TYPE <device> PARALLELISM <n>
```

where <n> is the parallelism value.

Using the RMAN SHOW command, you can view the RMAN configuration settings. If SHOW ALL is executed when connected to a target database, only node-specific configurations and database configurations are displayed.

You can return to the default value for any CONFIGURE command by executing the same command with the CLEAR option.

Using a Media Manager



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using a Media Manager

To use tape storage for your database backups, RMAN requires Oracle Secure Backup or a media manager.

A media manager is a utility that loads, labels, and unloads sequential media (such as tape drives) for the purpose of backing up, restoring, and recovering data. The Oracle database server calls Media Management Library (MML) software routines to back up and restore data files to and from media that is controlled by the media manager.

Note that the Oracle database server does not need to connect to the MML software when it backs up to disk.

Oracle Backup Solutions Program (BSP) provides a range of media management products that are compliant with Oracle's MML specification. Software that is compliant with the MML interface enables an Oracle database session to back up data to a media manager and request the media manager to restore backups. Check with your media vendor to determine whether it is a member of Oracle BSP.

Before you can begin using RMAN with a media manager, you must install the media manager software and make sure that RMAN can communicate with it. Instructions for this procedure should be available in the media manager vendor's software documentation.

Using a Media Manager (continued)

Depending on the product that you are installing, perform the following basic steps:

1. Install and configure the media management software on the target host or production network. No RMAN integration is required at this stage.
2. Ensure that you can make non-RMAN backups of operating system files on the target database host. This step makes it easier to troubleshoot problems at a later time. Refer to your media management documentation to learn how to back up files to the media manager.
3. Obtain and install the third-party media management module for integration with the Oracle database. This module must contain the library loaded by the Oracle database server when accessing the media manager.

Backup and Restore Operations Using a Media Manager

The following Recovery Manager script performs a data file backup to a tape drive controlled by a media manager:

```
run {
  # Allocating a channel of type 'sbt' for serial device
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt;
  BACKUP DATAFILE 3;
}
```

When Recovery Manager executes this command, it sends the backup request to the Oracle database session performing the backup. The Oracle database session identifies the output channel as a media management device and requests the media manager to load a tape and write the output.

The media manager labels and keeps track of the tape and the names of the files on each tape. The media manager also handles restore operations. When you restore a file, the following steps occur:

1. The Oracle database server requests the restoration of a particular file.
2. The media manager identifies the tape containing the file and reads the tape.
3. The media manager passes the information back to the Oracle database session.
4. The Oracle database server writes the file to disk.

Specifying a Backup Destination

Backups can be written to:

- Disk directory

```
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO DISK;
```

- Tape, using Oracle Secure Backup
- Media Management Library
 - Tape

```
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO TAPE;
```

- Disk or tape, using proxy copy
- Fast Recovery Area: Disk area set aside for backup and recovery and flashback database purposes
 - Define the location and the size.
 - Files are automatically retained and deleted as necessary.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying a Backup Destination

Backups can be written to a designated disk directory, a media management library (MML), or the Fast Recovery Area. Specifying a disk directory or the Fast Recovery Area means that backups go to hard-disk media. Typically, backups are regularly moved offline to tape via the media management interface in order to maintain disk space availability. Any disk directory can be specified as the destination of a backup provided that it already exists.

A media management library can be used to copy files to tape devices, or to carry out proxy copies. A proxy copy is where the MML is requested to make a copy of a file to a disk or tape device. The MML must be able to provide the proxy copy service for this to work.

If you set up a Fast Recovery Area, many backup and recovery tasks are simplified for you. The Oracle database automatically names files for you, and deletes obsolete files when there is space pressure.

To specify that backups are to be written to disk, use the first command in the slide.

Subsequently, when backups are made, if the `FORMAT` keyword is used (that specifies a disk directory location for the backup), then the backup is written there. If there is a Fast Recovery Area configured, then it goes there; otherwise, backups are written to a platform-specific default location.

To specify that a tape device is to be used, use the second command in the slide.

Note: See the *Oracle Secure Backup Administrator's Guide* for more information about Oracle Secure Backup.

Configuring and Allocating Channels

- Configure automatic channels with the `CONFIGURE` command:

```
RMAN> CONFIGURE DEVICE TYPE sbt;  
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO sbt;  
RMAN> CONFIGURE CHANNEL DEVICE TYPE sbt ...  
RMAN> BACKUP DATABASE;
```

- Allocate channels manually with the `ALLOCATE CHANNEL` command within a `RUN` block:

```
RMAN> RUN  
{  
  ALLOCATE CHANNEL ch1 DEVICE TYPE DISK;  
  BACKUP DATABASE PLUS ARCHIVELOG;  
}
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring and Allocating Channels

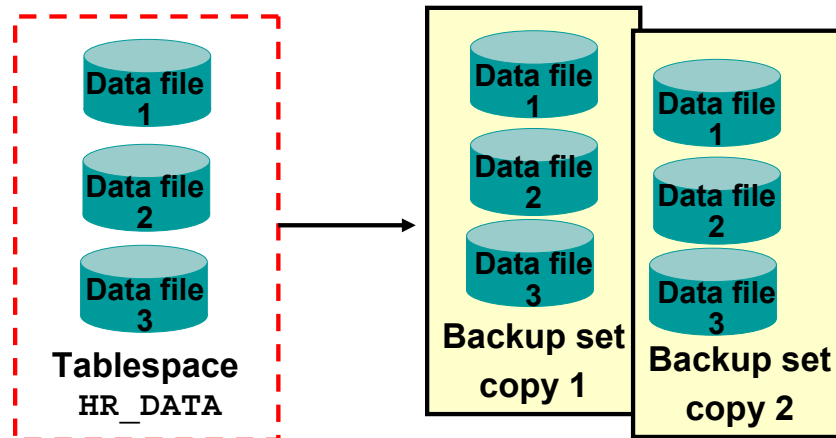
Choose from the following options for configuring channels and executing backups:

- Configure automatic channels with the `CONFIGURE` command, and then issue the `BACKUP` command at the `RMAN` prompt or within a `RUN` block.
- Manually allocate channels with the `ALLOCATE CHANNEL` command within a `RUN` block, and then issue `BACKUP` commands.

Creating Duplexed Backup Sets

To create a duplexed backup set, use:

- `CONFIGURE . . . BACKUP COPIES`
- `BACKUP . . . COPIES`



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Duplexed Backup Sets

RMAN can make up to four copies of a backup set simultaneously, each an exact duplicate of the others. A copy of a backup set is a copy of each backup piece in the backup set, with each copy getting a unique copy number (for example, `0tcm8u2s_1_1` and `0tcm8u2s_1_2`).

In most cases, the easiest method of duplexing backup sets is to use `BACKUP . . . COPIES` or `CONFIGURE . . . BACKUP COPIES` to duplex backup sets. For `DISK` channels, specify multiple values in the `FORMAT` option to direct the multiple copies to different physical disks. For `sbt` channels, if you use a media manager that supports Version 2 of the SBT API, then the media manager automatically puts each copy onto a separate medium (for example, a separate tape).

Note: The System Backup to Tape (SBT) API is the interface defined for Media Management Library (MML) developers, so that they can provide MMLs that communicate with RMAN.

Note that it is not possible to duplex backup sets to the Fast Recovery Area, and that duplexing applies only to backup sets, not image copies. You receive an error if you specify the `BACKUP . . . COPIES` option when creating image copy backups. The `CONFIGURE . . . BACKUP COPIES` setting is ignored for image copy backups.

Duplexed backup sets are typically used for tape backups.

Creating Duplexed Backup Sets Using CONFIGURE BACKUP COPIES

```
RMAN> CONFIGURE ARCHIVELOG BACKUP COPIES  
2> FOR DEVICE TYPE sbt TO 2;  
RMAN> CONFIGURE DATAFILE BACKUP COPIES  
2> FOR DEVICE TYPE sbt TO 2;  
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;  
RMAN> BACKUP DEVICE TYPE DISK AS COPY DATABASE;
```

Two copies of the backup are made to two different tapes.

Not affected by the COPIES configuration setting. Only one copy is made on disk.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Duplexed Backup Sets Using CONFIGURE BACKUP COPIES

Use the CONFIGURE . . . BACKUP COPIES command to specify the number of identical backup sets that you want to create on the specified device type. This setting applies to all backups except control file autobackups (because the autobackup of a control file always produces one copy) and backup sets when backed up with the BACKUP BACKUPSET command.

Note: You must have automatic channels configured.

To create a duplexed backup set with CONFIGURE BACKUP COPIES, perform the following steps:

1. Configure the number of copies on the desired device type for data files and archived redo log files.
2. Execute the BACKUP command.
3. Issue a LIST BACKUP command to verify your backup.

Note: The last BACKUP command is not affected by the COPIES configuration setting. It creates a single copy to disk.

Backup Optimization

- Skips already backed-up files
- Is used when:
 - Backup optimization is enabled

```
RMAN> CONFIGURE BACKUP OPTIMIZATION ON;
```

- BACKUP DATABASE, BACKUP ARCHIVELOG with ALL or LIKE options, or BACKUP BACKUPSET ALL commands are executed
 - Only one type of channel is allocated
- Can be overridden with the FORCE option

```
RMAN> BACKUP DEVICE TYPE sbt BACKUPSET ALL FORCE;
```

- Is always used for RECOVERY AREA, DB_RECOVERY_FILE_DEST, and RECOVERY FILES backup options

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Backup Optimization

If you enable backup optimization, the BACKUP command skips files when the identical files have already been backed up to the specified device type.

If RMAN determines that a file is identical and it has already been backed up, it is a candidate to be skipped. However, RMAN performs further checking to determine whether to skip the file, because both the retention policy and the backup duplexing feature are factors in the algorithm that RMAN uses to determine whether there are sufficient backups on the specified device type.

Refer to the *Oracle Database Backup and Recovery User's Guide* for detailed information about the criteria that RMAN uses to determine whether a file is identical and the backup optimization algorithm.

You can enable backup optimization on the Backup Settings page in Enterprise Manager or by issuing the CONFIGURE BACKUP OPTIMIZATION ON command. By default, backup optimization is disabled.

Backup optimization is automatically enabled for the BACKUP RECOVERY AREA | DB_RECOVERY_FILE_DEST and BACKUP RECOVERY FILES commands.

Configuring Backup Optimization (continued)

To override backup optimization and back up all files whether or not they have changed, specify the **FORCE** option on the **BACKUP** command as in the following example:

```
BACKUP DEVICE TYPE sbt BACKUPSET ALL FORCE;
```

Note that the **FORCE** option does not apply to files in the recovery area.

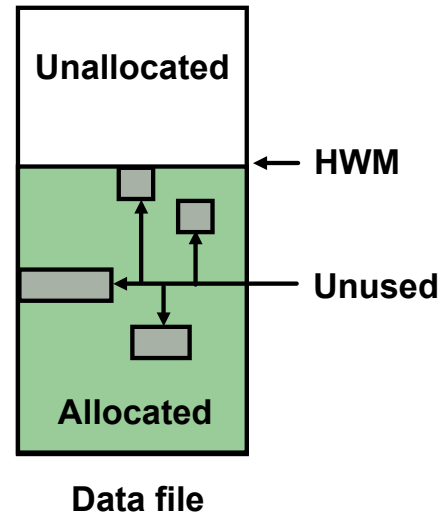
You can disable backup optimization on a persistent basis using Enterprise Manager or by issuing the following command:

```
CONFIGURE BACKUP OPTIMIZATION OFF;
```

Saving Backup Space with Unused Block Compression

The following blocks may be skipped during certain types of backup operations:

- Unallocated blocks: These are above the data file's high-water mark (HWM).
- Unused blocks: These are blocks that have been allocated but no longer belong to a segment.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Saving Backup Space with Unused Block Compression

When certain types of backups occur, RMAN is able to skip some blocks. Unallocated blocks may be skipped. They are those that have not been allocated; they are above the HWM. Also, some allocated blocks that no longer belong to a segment (are not in use) may be skipped, provided the following are true:

- There are no guaranteed restore points defined.
- The data file contains data only for locally managed tablespaces.
- The data file is being backed up to a backup set as part of a full backup or a level 0 incremental.
- The backup is going to disk or Oracle Secure Backup is the media manager.

Compressing Backups

RMAN can perform binary compression on any backup set that is generated.

- It can be performed in addition to unused block compression.
- Available compression algorithms are: HIGH, MEDIUM, LOW, and BASIC.
- No extra steps are required by the DBA to restore a compressed backup.

```
CONFIGURE COMPRESSION ALGORITHM 'HIGH/MEDIUM/LOW/BASIC'
```

```
run {  
SET COMPRESSION ALGORITHM 'HIGH/MEDIUM/LOW/BASIC';  
..  
}
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Compressing Backups




Undo data that is not needed for transaction recovery (for example, for committed transactions), is not backed up. The benefit is reduced overall backup time and storage by not backing up undo that applies to committed transactions. This optimization is automatically enabled.

While unused block compression decreases the number of blocks that are written to the backup (and the backup time), binary compression can be used to algorithmically compact the data that is written. The available compression algorithms are HIGH, MEDIUM, LOW, and BASIC. If you specify it for a specific backup device, then use the COMPRESSED keyword after the BACKUP TYPE TO clause.

You do not have to perform any additional steps when restoring a compressed backup. Note, however, that compression and decompression operations require CPU resources. So both creating and restoring a compressed backup will, of course, probably take longer and require more system resources.

When choosing an algorithm, consider your disk space in addition to dynamic system resources such as CPU and memory.

Using RMAN Backup Compression

Compression Ratio or Level	Considerations	Requires Advanced Compression Option
LOW	Fastest. Best suited to address backup: CPU resources	
MEDIUM	Fast. Good balance of CPU usage and compression ratio	
HIGH	Best compression ratio at the expense of high CPU consumption. Best suited to address backup constraint: network.	
BASIC	Fair. Compression ratio similar to MEDIUM at expense of additional CPU usage. Compression ratio between MEDIUM and HIGH	

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using RMAN Backup Compression

Binary compression of backup sets is supported with the algorithm settings as shown in the slide. All modes except BASIC require the Oracle Advanced Compression Database option.

Because the performance of the various compression levels depends on the nature of the data in the database, network configuration, system resources, and the type of your computer system and its capabilities, Oracle Corporation cannot document universally applicable performance statistics. To decide which level is best for you, consider how balanced your system is regarding bandwidth into the CPU, as well as the actual speed of the CPU. It is highly recommended that you run tests with the different compression levels on the data in your environment. Choosing a compression level based on your own environment, network traffic (workload), and dataset is the only way to ensure that the backup set compression level can satisfy your organization's performance requirements and any applicable service-level agreements.

The following level or compression ratios are available:

- LOW: This level is the fastest. It provides less compression than MEDIUM, but uses the least CPU. (It corresponds to the LZO compression.)
- MEDIUM: This level provides a good balance of CPU usage and compression ratio. (It corresponds to the ZLIB compression.)
- HIGH: This level provides the best compression ratio, but consumes the most CPU. (It corresponds to the GZIP compression.)
- BASIC: This corresponds to BZIP2 (10g style compression).

Encrypting Backups

- **Transparent encryption:** With a wallet (default)
- **Password encryption:** With a password (no wallet)
- **Dual mode encryption:**
 - Has both transparent and password encryption modes
 - Can be restored in either transparent or password mode



Copyright © 2009, Oracle. All rights reserved.

Encrypting Backups

You can encrypt backups in one of three ways:

- **Transparent encryption:** This method uses a wallet, and it is the default mode.
- **Password encryption:** This method of encryption relies on a password. There is no need to configure a wallet. You must know the password that was used for the backup in order to restore.
- **Dual mode encryption:** Both transparent and password encryption are used. In order to restore, either the transparent mode or the password mode can be used. This type of encryption is useful if you usually restore your backups to the local site, but sometimes ship the backups to other sites.

Encrypting backups is covered in detail in the *Oracle Database 11g: Security* course.

Quiz

How can you examine the persistent RMAN settings for a database? Select all true answers:

1. Connected only to the target, you enter `SHOW ALL` at the RMAN prompt.
2. In a SQL*Plus session, you use the `SHOW RMAN CONFIGURATION` command.
3. Connected only to the recovery catalog, you enter `SHOW ALL` at the RMAN prompt.
4. Logged in to the target database instance, you query the `V$RMAN_CONFIGURATION` view.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 4

Quiz

Select the true statements about RMAN backup functionality:

1. Backup `FORCE` overrides the backup optimization and backs up all files, whether they have changed or not.
2. Persistent RMAN settings can only be used for one-time backups.
3. Parallelism is the number of possible streams of data to and from a backup device .

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 3

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager to configure backup settings
- Enable control file autobackup
- Configure backup destinations
- Allocate channels for tape destination
- Configure backup optimization
- Create a compressed backup
- Create an encrypted backup

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 4 Overview: Configuring Backup Specifications

This practice covers the following topics:

- Configuring RMAN persistent settings
- Configuring autobackup for control file

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring Backup Specifications

Practice Tip: Because the output of the RMAN commands can be quite long, consider using the RMAN SPOOL LOG command to direct the output to your specified file.

Example

```
RMAN> SPOOL LOG TO '/home/oracle/labs/my_lab_output.txt';
```


5

Creating Backups with RMAN

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

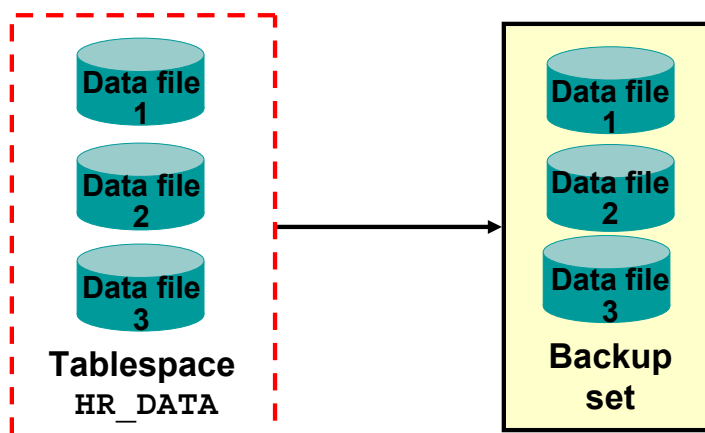
- Create image file backups
- Create a whole database backup
- Create a full database backup
- Enable fast incremental backup
- Create duplex backup sets
- Back up a backup set
- Create RMAN multi-section backup
- Create an archival backup for long-term retention
- Report on and maintain backups

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Backup Sets

```
RMAN> BACKUP AS BACKUPSET  
2> FORMAT '/BACKUP/df_%d_%s_%p.bus'  
3> TABLESPACE hr_data;
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

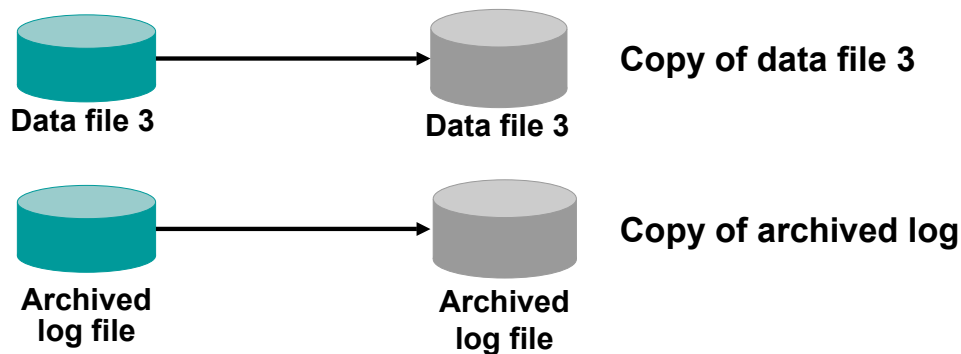
Creating Backup Sets

RMAN can store its backups in an RMAN-exclusive format called a backup set. A backup set is a collection of files called backup pieces, each of which may contain a backup of one or more database files.

Note: The `FORMAT` parameter specifies a pattern to use in creating a file name for the backup pieces created by this command. The `FORMAT` specification can also be provided through the `ALLOCATE CHANNEL` and `CONFIGURE` commands.

Creating Image Copies

```
RMAN> BACKUP AS COPY DATAFILE '/ORADATA/users_01_db01.dbf';  
RMAN> BACKUP AS COPY ARCHIVELOG LIKE '/arch%';
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Image Copies

An image copy is a clone of a single data file, archived redo log, or control file. An image copy can be created with the `BACKUP AS COPY` command or with an operating system command. When you create the image copy with the `RMAN BACKUP AS COPY` command, the server session validates the blocks in the file and records the copy information in the control file.

An image copy has the following characteristics:

- An image copy can be written only to disk. When large files are being considered, copying may take a long time, but restoration time is reduced considerably because the copy is available on the disk.
- If files are stored on disk, they can be used immediately by using the `SWITCH` command in RMAN, which is equivalent to the `ALTER DATABASE RENAME FILE SQL` statement.
- In an image copy, all blocks are copied, whether they contain data or not, because an Oracle database process copies the file and performs additional actions such as checking for corrupt blocks and registering the copy in the control file. To speed up the process of copying, you can use the `NOCHECKSUM` parameter. By default, RMAN computes a checksum for each block backed up, and stores it with the backup. When the backup is restored, the checksum is verified. For more information about the `NOCHECKSUM` option of the `BACKUP` command, see the *Oracle Database Backup and Recovery Reference*.

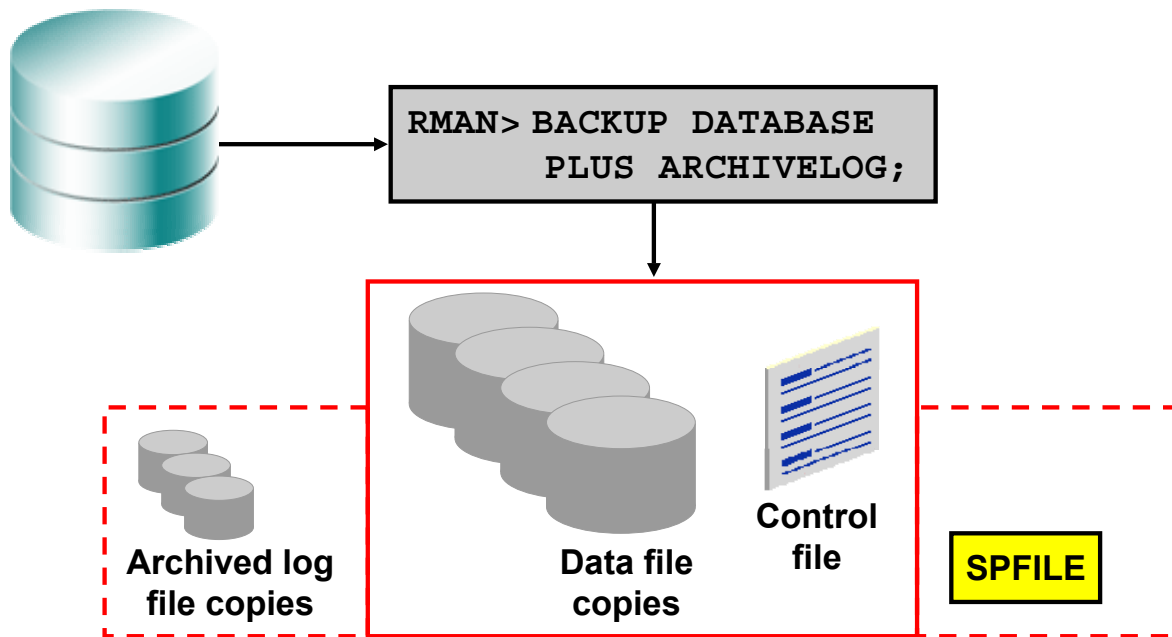
Creating Image Copies (continued)

- An image copy can be part of a full or incremental level 0 backup because a file copy always includes all blocks. You must use the level 0 option if the copy will be used in conjunction with an incremental backup set.

The example in the slide creates the following image copies:

- A copy of the /ORADATA/users01_db01.dbf data file
- A copy of the archived redo log files

Creating a Whole Database Backup



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating a Whole Database Backup

A whole database backup can be either backup sets or image copies of the entire set of data files and must include the control file. You can optionally include the server parameter file (SPFILE) and archived redo log files. Using Recovery Manager (RMAN) to make an image copy of all the database files simply requires mounting or opening the database, starting RMAN, and entering the BACKUP command shown in the slide. Optionally, you can supply the DELETE INPUT option when backing up archivelog files. That causes RMAN to remove the archivelog files after backing them up. This is useful especially if you are not using a Fast Recovery Area, which would perform space management for you, deleting files when space pressure grows. In that case, the command in the slide would look like this:

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

You must have issued the following CONFIGURE commands to make the backup as described previously:

- CONFIGURE DEFAULT DEVICE TYPE TO disk;
- CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY;
- CONFIGURE CONTROLFILE AUTOBACKUP ON;

You can also create a backup (either a backup set or image copies) of previous image copies of all data files and control files in the database by using the following command:

```
RMAN> BACKUP COPY OF DATABASE;
```

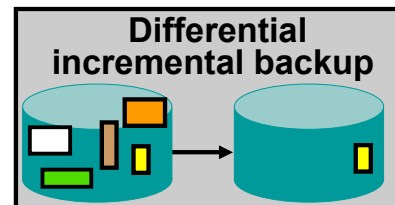
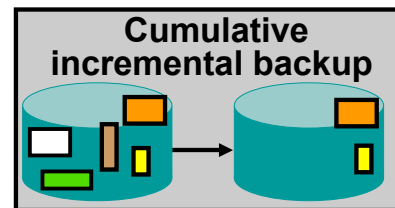
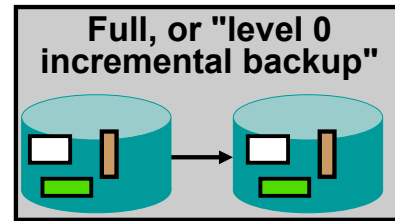
Creating a Whole Database Backup (continued)

By default, RMAN executes each BACKUP command serially. However, you can parallelize the copy operation by:

- Using the `CONFIGURE DEVICE TYPE DISK PARALLELISM n` command, where `n` is the desired degree of parallelism
- Allocating multiple channels
- Specifying one `BACKUP AS COPY` command and listing multiple files

RMAN Backup Types

- A full backup contains all used data file blocks.
- A level 0 incremental backup is equivalent to a full backup that has been marked as level 0.
- A cumulative level 1 incremental backup contains only blocks modified since the last level 0 incremental backup.
- A differential level 1 incremental backup contains only blocks modified since the last incremental backup.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

RMAN Backup Types

Full Backups

A full backup is different from a whole database backup. A full data file backup is a backup that includes every used data block in the file. RMAN copies all blocks into the backup set or image copy, skipping only those data file blocks that have never been used. For a full image copy, the entire file contents are reproduced exactly. A full backup cannot be part of an incremental backup strategy; it cannot be the parent for a subsequent incremental backup.

Incremental Backups

An incremental backup is either a level 0 backup, which includes every block in the data files except blocks that have never been used, or a level 1 backup, which includes only those blocks that have been changed since a previous backup was taken. A level 0 incremental backup is physically identical to a full backup. The only difference is that the level 0 backup (as well as an image copy) can be used as the base for a level 1 backup, but a full backup can never be used as the base for a level 1 backup.

Incremental backups are specified using the `INCREMENTAL` keyword of the `BACKUP` command. You specify `INCREMENTAL LEVEL [0 | 1]`.

RMAN Backup Types (continued)

RMAN can create multilevel incremental backups as follows:

- **Differential:** Is the default type of incremental backup that backs up all blocks changed after the most recent incremental backup at either level 1 or level 0
- **Cumulative:** Backs up all blocks changed after the most recent backup at level 0

Examples

- To perform an incremental backup at level 0, use the following command:
`RMAN> BACKUP INCREMENTAL LEVEL 0 DATABASE;`
- To perform a differential incremental backup, use the following command:
`RMAN> BACKUP INCREMENTAL LEVEL 1 DATABASE;`
- To perform a cumulative incremental backup, use the following command:
`RMAN> BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;`

RMAN makes full backups by default if neither FULL nor INCREMENTAL is specified. Unused block compression causes never-written blocks to be skipped when backing up data files to backup sets, even for full backups.

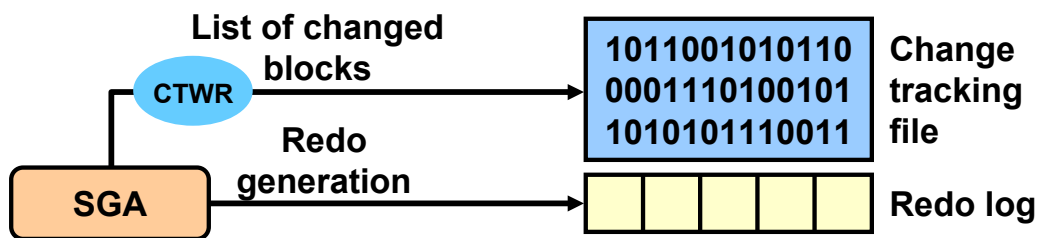
A full backup has no effect on subsequent incremental backups, and is not considered part of any incremental backup strategy, although a full image copy backup can be incrementally updated by applying incremental backups with the RECOVER command. This is covered in the lesson titled “Using RMAN to Perform Recovery.”

Note: It is possible to perform any type of backup (full or incremental) of a database that is in NOARCHIVELOG mode—if, of course, the database is not open. Note also that recovery is limited to the time of the last backup. The database can be recovered to the last committed transaction only when the database is in ARCHIVELOG mode.

Fast Incremental Backup

Implemented by block change tracking, which:

- Maintains a record of what blocks have changed since the last backup
- Writes this record to a file, as redo is generated
- Is automatically accessed when a backup is done, making the backup run faster



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Fast Incremental Backup

The goal of an incremental backup is to back up only those data blocks that have changed since a previous backup. You can use RMAN to create incremental backups of data files, tablespaces, or the whole database. When making an incremental backup, RMAN reads only those blocks referenced to locate the changed blocks since the last backup. That makes the backup smaller because only changed blocks are backed up. It also makes recovery faster because fewer blocks need to be restored.

You can perform fast incremental backup by enabling block change tracking. Block change tracking writes to a file the physical address of each block that is changed. When it is time to perform the incremental backup, RMAN can look at the block change tracking file, and back up only those blocks referenced there; it does not have to scan every block to see if it has changed since the last backup. This makes the incremental backup faster.

The maintenance of the tracking file is fully automatic and does not require your intervention. The size of the block change tracking file is proportional to the:

- Database size, in bytes
- Number of enabled threads in a RAC environment
- Number of old backups maintained by the block change tracking file

The minimum size for the block change tracking file is 10 MB, and any new space is allocated in 10 MB increments. The Oracle database does not record block change information by default.

Enabling Fast Incremental Backup

ORACLE Enterprise Manager 10g Database Control

Setup Preferences Help Logout Database

Database Instance: orcl > Backup Settings

Backup Settings

Device Backup Set Policy

Backup Policy

☐ Automatically backup the control file and server parameter file (SPFILE) with every backup and database structural change

Autobackup Disk Location
An existing directory or diskgroup name where the control file and server parameter file will be backed up. If you do not specify a location, the files will be backed up to the flash recovery area location.

☐ Optimize the whole database backup by skipping unchanged files such as read-only and offline datafiles that have been backed up

☐ Enable block change tracking for faster incremental backups

Block Change Tracking File
Specify a location and file, otherwise an Oracle managed file will be created in the database area.

```
ALTER DATABASE
{ENABLE|DISABLE} BLOCK CHANGE TRACKING
[USING FILE '...']
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Enabling Fast Incremental Backup

You enable block change tracking from the Database Control home page. Navigate to Availability > Backup Settings > Policy. You do not need to set the block change tracking file destination if the DB_CREATE_FILE_DEST initialization parameter is set because the file is created as an Oracle Managed File (OMF) file in the DB_CREATE_FILE_DEST location. You can, however, specify the name of the block change tracking file, placing it in any location you choose.

You can also enable or disable this feature by using an ALTER DATABASE command. If the change tracking file is stored in the database area with your database files, then it is deleted when you disable change tracking. You can rename the block change tracking file by using the ALTER DATABASE RENAME command. Your database must be in the MOUNT state to rename the tracking file. The ALTER DATABASE RENAME FILE command updates the control file to refer to the new location. You can use the following syntax to change the location of the block change tracking file:

```
ALTER DATABASE RENAME FILE '...' TO '...';
```

Note: RMAN does not support backup and recovery of the block change tracking file. For this reason, you should not place it in the Fast Recovery Area.

Monitoring Block Change Tracking

```
SQL> SELECT filename, status, bytes
2 FROM v$block_change_tracking;
```

```
SQL> SELECT file#, avg(datafile_blocks),
2          avg(blocks_read),
3          avg(blocks_read/datafile_blocks)
4          * 100 AS PCT_READ_FOR_BACKUP,
5          avg(blocks)
5 FROM v$backup_datafile
6 WHERE used_change_tracking = 'YES'
7 AND incremental_level > 0
8 GROUP BY file#;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring Block Change Tracking

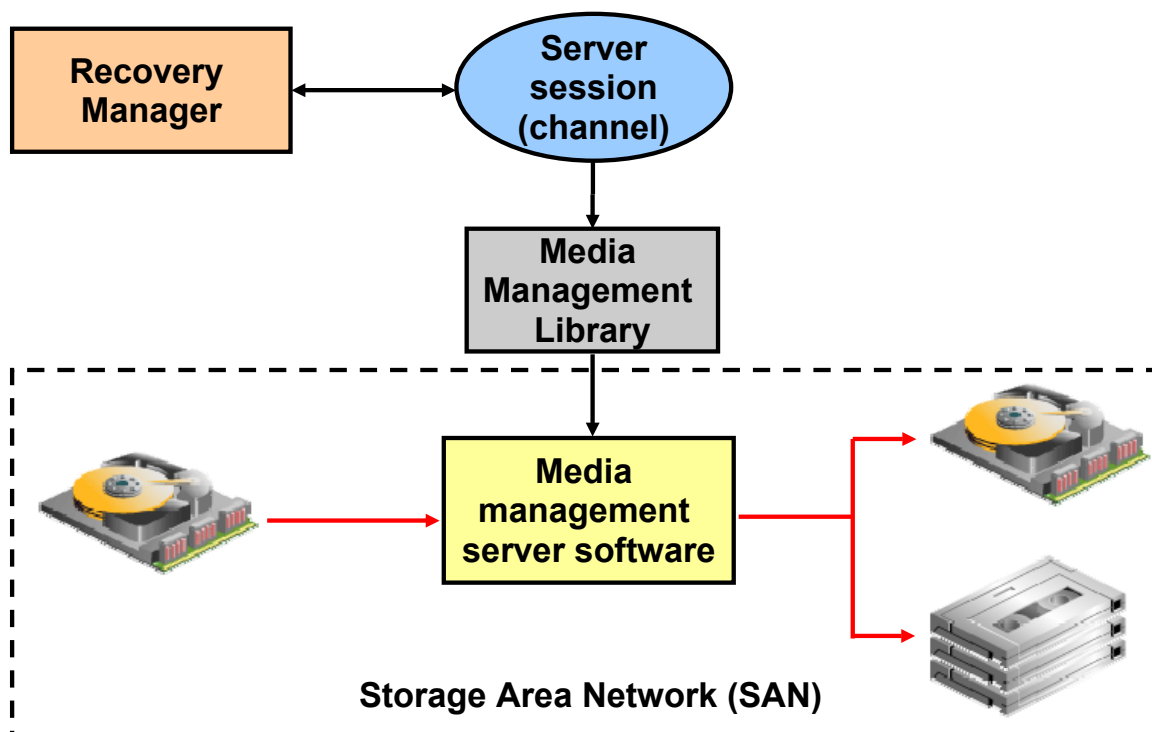
The output of the V\$BLOCK_CHANGE_TRACKING view shows where the block change tracking file is located, the status of block change tracking (ENABLED/DISABLED), and the size (in bytes) of the file.

The query on the V\$BACKUP_DATAFILE view shows how effective the block change tracking is in minimizing the incremental backup I/O (the PCT_READ_FOR_BACKUP column). A high value indicates that RMAN reads most blocks in the data file during an incremental backup. You can reduce this ratio by decreasing the time between the incremental backups.

A sample formatted output from the V\$BACKUP_DATAFILE query is shown below:

FILE#	BLOCKS_IN_FILE	BLOCKS_READ	PCT_READ_FOR_BACKUP	BLOCKS_BACKED_UP
1	56320	4480	7	462
2	3840	2688	70	2408
3	49920	16768	33	4457
4	640	64	10	1
5	19200	256	1	91

Performing Proxy Copies



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Proxy Copies

Use the PROXY option of the RMAN BACKUP command to request a MML to perform the copy of the files.

Syntax:

```
BACKUP [AS BACKUPSET] ... PROXY [ONLY] DATABASE|TABLESPACE....
```

The PROXY ONLY option is useful for those media managers and storage networks where having the backup done by proxy may substantially reduce the storage net traffic.

Some media management products can completely manage all data movement between Oracle data files and the backup devices. Some products that use high-speed connections between storage and media subsystems can reduce much of the backup load from the primary database server. This is beneficial in that the copying takes place across the SAN instead of the LAN. At that point, RMAN is no longer involved in the operation, except for communicating status across the LAN to and from the MML.

Creating Duplexed Backup Sets Using BACKUP COPIES

```
RMAN> BACKUP AS BACKUPSET DEVICE TYPE sbt  
2> COPIES 2  
3> INCREMENTAL LEVEL 0  
4> DATABASE;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Duplexed Backup Sets Using BACKUP COPIES

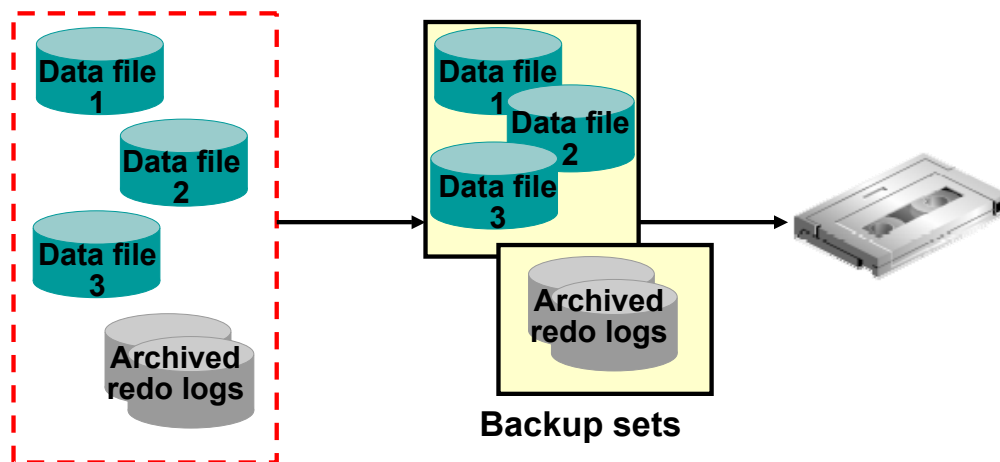
You can use the BACKUP command with the COPIES option to override other COPIES or DUPLEX settings to create duplexed backup sets.

To duplex a backup with BACKUP COPIES, perform the following steps:

1. Specify the number of identical copies with the COPIES option of the BACKUP command.
2. Issue a LIST BACKUP command to verify your backup.

Creating Backups of Backup Sets

```
RMAN> BACKUP DEVICE TYPE DISK AS BACKUPSET  
2> DATABASE PLUS ARCHIVELOG;  
RMAN> BACKUP DEVICE TYPE sbt BACKUPSET ALL;
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Backups of Backup Sets

Use the RMAN BACKUP BACKUPSET command to back up previously created backup sets. Only backup sets that were created on device type DISK can be backed up using RMAN. The backup sets can be backed up to any available device type.

The BACKUP BACKUPSET command uses the default disk channel to copy backup sets from disk to disk. To back up from disk to tape, you must either configure or manually allocate a nondisk channel.

Backing Up Read-Only Tablespaces

Considerations for backing up read-only tablespaces:

- Backup optimization causes RMAN to back up read-only tablespaces only when there does not exist a backup that satisfies the retention policy.
- If you change the tablespace to read/write, back it up immediately.
- You can use the `SKIP READONLY` option of the RMAN `BACKUP` command to skip read-only tablespaces or data files.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

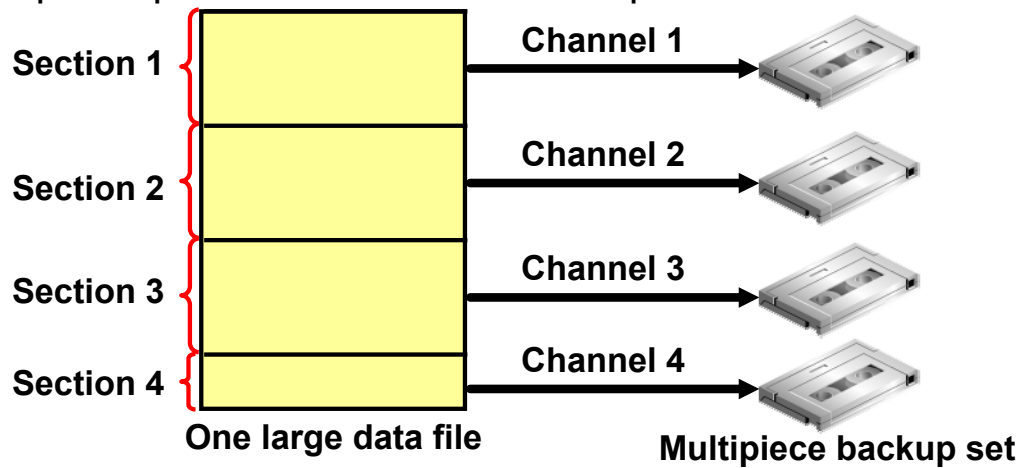
Backing Up Read-Only Tablespaces

Because read-only tablespaces are not being written to, there is no need to continually back them up as you do read/write tablespaces. You can use the `SKIP READONLY` option of the `BACKUP` command to let RMAN know to not back up read-only tablespaces.

Configuring Backup and Restore for Very Large Files

Multisection backups of a single file:

- Are created by RMAN, with your specified size value
- Are processed independently (serially or in parallel)
- Produce multipiece backup sets
- Improve performance of the backup



Copyright © 2009, Oracle. All rights reserved.

Configuring Backup and Restore for Very Large Files

Oracle data files can be up to 128 TB in size. Normally, the smallest unit of an RMAN backup is an entire file. This is not practical with such large files. RMAN can optionally break up large files into sections and back up and restore these sections independently. You do this by creating multisection backups, which break up the files generated for the backup set into separate files. This can be done only with backup sets, not image copies.

Each file section is a contiguous range of blocks of a file. Each file section can be processed independently, either serially or in parallel. Backing up a file into separate sections can improve the performance of the backup operation, and it also allows large file backups to be restarted.

A multisection backup job produces a multipiece backup set. Each piece contains one section of the file. All sections of a multisection backup, except perhaps for the last section, are of the same size. There are a maximum of 256 sections per file.

Note: You should not apply large values of parallelism to back up a large file that resides on a small number of disks, as that would defeat the purpose of the parallel operation; multiple simultaneous accesses to the same disk device would be competing with each other.

This feature is built into RMAN. No installation is required beyond the normal installation of Oracle Database 11g. COMPATIBLE must be set to at least 11.0, because earlier releases cannot restore multisection backups.

Creating RMAN Multisection Backups

RMAN command syntax:

```
BACKUP <options> SECTION SIZE <integer> [K | M | G]
```

```
VALIDATE DATAFILE <options> SECTION SIZE <integer> [K | M | G]
```

Example:

```
RMAN> BACKUP DATAFILE 5 SECTION SIZE = 25M TAG 'section25mb';  
backing up blocks 1 through 3200  
  
piece handle=/u01/.../o1_mf_nnndf_SECTION25MB_382dryt4_.bkp  
tag=SECTION25MB comment=NONE  
  
...  
backing up blocks 9601 through 12800  
  
piece handle=/u01/.../o1_mf_nnndf_SECTION25MB_382dsto8_.bkp  
tag=SECTION25MB comment=NONE
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating RMAN Multisection Backups

The BACKUP and VALIDATE DATAFILE commands accept the following option:

SECTION SIZE <integer> [K | M | G]

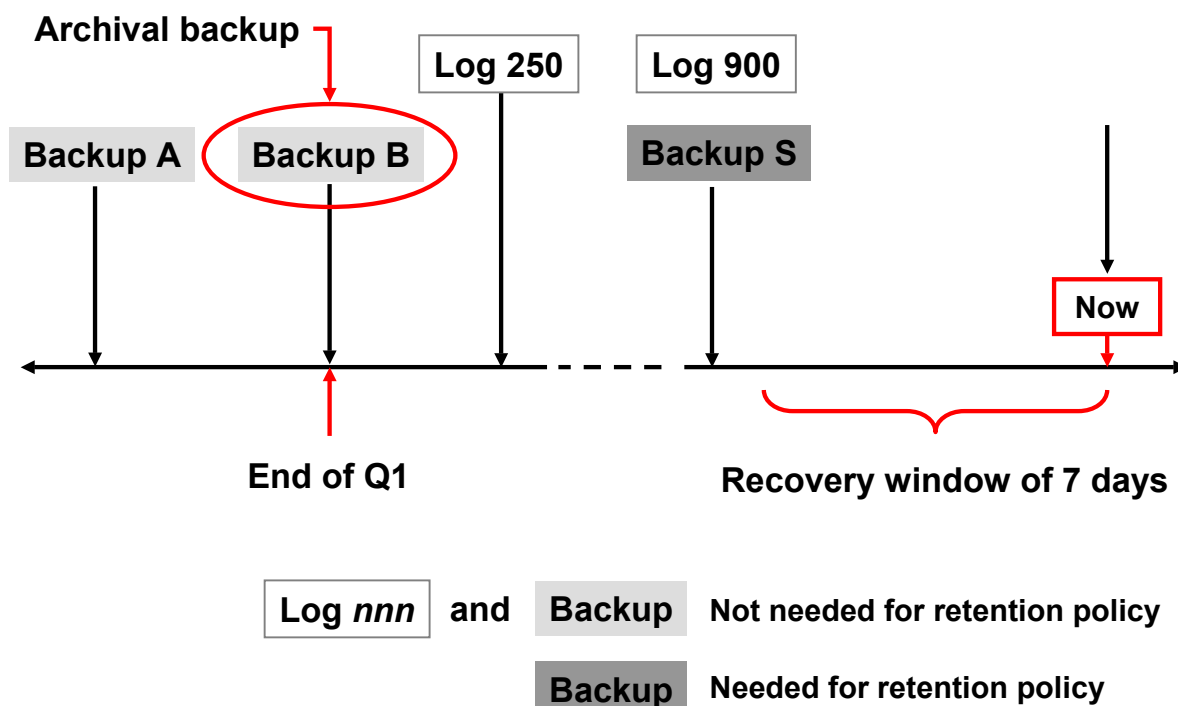
Use this to specify your planned size for each backup section. The option is both a backup command- and backup spec-level option, so that you can apply different section sizes to different files in the same backup job.

In the example in the slide, a backup of data file 5 is being taken, and the section size is specified as 25 MB. The data file is 100 MB in size, so four sections are created. Note that, as indicated by the block ranges, block contiguity is maintained as they are written to the section files.

Viewing Metadata About Your Multisection Backup

- The V\$BACKUP_SET and RC_BACKUP_SET views have a MULTI_SECTION column, which indicates whether this is a multisection backup or not.
- The V\$BACKUP_DATAFILE and RC_BACKUP_DATAFILE views have a SECTION_SIZE column, which specifies the number of blocks in each section of a multisection backup. Zero means a whole-file backup.

Archival Backups: Concepts



Copyright © 2009, Oracle. All rights reserved.

Archival Backups: Concepts

If you need to preserve an online backup for a specified amount of time, RMAN normally assumes you might want to perform point-in-time recovery for any time since that backup to the present. To satisfy this scenario, RMAN keeps the archived logs for that time period. However, you may have a requirement to simply keep the specific backup (and what is necessary to keep it consistent and recoverable) for a specified amount of time—for example, for two years. You do not have the intention of recovering to a point in time since that backup, but you just want to be able to recover to the exact time of the backup, and no later. You also want to maintain a retention policy that keeps your backup area free of clutter, so making it reach back two years is not acceptable. This is a common need, when meeting business or legal requirements for data retention.

An archival backup solves this problem. If you mark a backup as an archival backup, that attribute overrides any configured retention policy for the purpose of this backup. You can retain archival backups such that they are either considered obsolete only after a specific time that you specify, or never considered obsolete. If you want to specify the latter, you need to use a recovery catalog.

The `KEEP` clause creates an archival backup that is a snapshot of the database at a point in time. The only redo logs that are kept are those required to restore this backup to a consistent state. The `RESTORE POINT` clause issued after the backup is completed determines the number of redo logs that are kept (enough to restore the backup to the `RESTORE POINT` time).

Archival Backups: Concepts (continued)

An archival backup also guarantees that all of the files needed to restore the backup are included. RMAN includes the data files, `SPFILE`, archived log files (only those needed to recover an online backup), and the relevant autobackup files. All these files must go to the same media family (or group of tapes).

You can also specify a restore point to be created, which has the same SCN as the archival backup. That essentially gives a meaningful name to the point of time the backup was made.

After an archival backup is created, it is retained for as long as specified. Even if you have a much smaller retention window and run the `DELETE OBSOLETE` command, the archival backup remains.

This backup is a snapshot of the database at a point in time, and can be used to restore the database to another host for testing purposes, for example.

Note: Archival backups cannot be written to the Fast Recovery Area. So if you have one, you must provide a `FORMAT` clause to specify a different location.

Creating Archival Backups with EM

Schedule Customized Backup: Settings

Database: **orcl.us.oracle.com** [Cancel] [Back] Step 3 of 5 [Next]

Backup Strategy: **Customized Backup**

Object Type: **Tablespaces**

These are the settings for your current backup job. You can select your backup destination directly from this page. You can also view the default settings or override the settings by clicking the buttons below.

☒ Disk

Disk Backup Location: **/archive_bu**

☐ Tape

Media Management Vendor(MMV) Library Parameters: **not specified**

[View Default Settings] [Override Current Settings]

Changed settings will only apply to the current backup job.

Override Retention Policy

☐ Do not override the retention policy

☒ Keep this backup for the specified number of days Days: **365**

[Device] [Backup Set] [Policy]

[Cancel] [OK]

Creating Archival Backups with EM

To create an archival backup using Enterprise Manager, perform the following steps:

1. Select Availability > Schedule Backup > Schedule Customized Backup.
2. Follow the steps of the Schedule Customized Backup wizard until you are on the Settings page.
3. Click Override Current Settings and then the Policy tab. In the Override Retention Policy section, you can select to keep a backup for a specified number of days. A restore point is generated based on the backup job name. You probably also want to specify a different destination for the backup files; to do this, use the Device tab.

Backups created with the KEEP option include the SPFILE, control files, and archive redo log files required to restore this backup, and data files. This backup is a snapshot of the database at a point in time, and can be used to restore the database to another host.

Creating Archival Backups with RMAN

- Specifying the `KEEP` clause when the database is online includes both data file and archive log backup sets:

```
KEEP {FOREVER | UNTIL TIME [=] ' date_string '}
NOKEEP
[RESTORE POINT rsname]
```

- List all restore points known to the RMAN repository:

```
LIST RESTORE POINT ALL;
```

- Display a specific restore point:

```
LIST RESTORE POINT 'rsname';
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Archival Backups with RMAN

Use the following syntax to create an archival backup using RMAN:

```
BACKUP ... KEEP {FOREVER|UNTIL TIME 'SYSDATE + <n>'} RESTORE POINT
<restore_point_name>
```

The `UNTIL TIME` clause enables you to specify when the archival backup is no longer immune to the retention policy. You can optionally specify `FOREVER`, meaning that the backup is an archival backup until you take some other action to change that.

Optionally, use the `RESTORE POINT` clause to specify the name of a restore point to be associated with this backup. The `RESTORE POINT` clause creates a “consistency” point in the control file. It assigns a name to a specific SCN. The SCN is captured just after the data file backup completes. The archival backup can be restored and recovered for this point in time, enabling the database to be opened. In contrast, the `UNTIL TIME` clause specifies the date until which the backup must be kept.

Managing Archival Database Backups

1 Archiving a database backup:

```
RMAN> CONNECT TARGET /  
RMAN> CONNECT CATALOG rman/rman@catdb  
RMAN> CHANGE BACKUP TAG 'consistent_db_bkup'  
2> KEEP FOREVER;
```

2 Changing the status of a database copy:

```
RMAN> CHANGE COPY OF DATABASE CONTROLFILE NOKEEP;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Managing Archival Database Backups

The **CHANGE** command changes the exemption status of a backup or copy in relation to the configured retention policy. For example, you can specify **CHANGE . . . NOKEEP**, to make a backup that is currently exempt from the retention policy eligible for the **OBSOLETE** status.

The first example changes a consistent backup into an archival backup, which you plan to store offsite. Because the database is consistent and, therefore, requires no recovery, you do not need to save archived redo logs with the backup.

The second example specifies that any long-term image copies of data files and control files should lose their exempt status and so become eligible to be obsolete according to the existing retention policy. This statement essentially removes the archival attribute from those backup files. If you do not specify a tag, as in this case, then the **CHANGE** execution applies to all backups of the type specified. You should specify a tag to change only the backup files you intend to change.

Note: The **RESTORE POINT** option is not valid with the **CHANGE** command, because there is no way to create the restore point for a time that has already passed (when the backup was created).

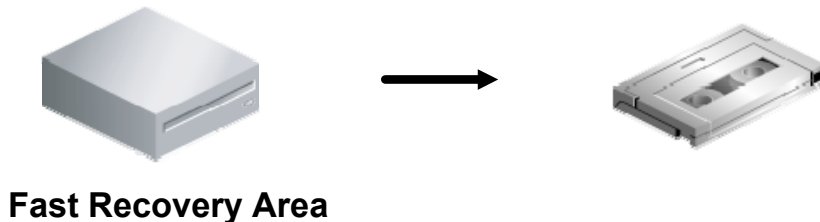
Backing Up Recovery Files

- Back up only the files in the Fast Recovery Area:

```
RMAN> BACKUP RECOVERY AREA
```

- Back up all recovery files:

```
RMAN> BACKUP RECOVERY FILES
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Backing Up Recovery Files

There are two ways to back up recovery data. The `BACKUP RECOVERY AREA` command backs up all files that are found in the current and any previous Fast Recovery Areas. The `BACKUP RECOVERY FILES` command backs up all recovery files, even if they are not in the FRA. You gain added protection from loss by using the latter, which would back up, for example, any copies of control files or data files that are not in the Fast Recovery Area.

By default, backup optimization is in effect for these two commands, even if you have disabled it using the `CONFIGURE` command. This means that the only recovery files that this command backs up are those that are not already backed up. You can force all files to be backed up by using the `FORCE` option.

You cannot specify `DEVICE TYPE DISK` for either of these commands.

Note: RMAN backs up only database files: data files, control files, SPFILES, archive log files, and backups of these files. Placing an operating system file in the Fast Recovery Area causes it to be included with a backup of the recovery area.

Managing Backups: Reporting

Use the following RMAN commands to obtain information about your backups:

- **LIST:** Displays information about backup sets, proxy copies, and image copies recorded in the repository
- **REPORT:** Produces a detailed analysis of the repository
- **REPORT NEED BACKUP:** Lists all data files that require a backup
- **REPORT OBSOLETE:** Identifies files that are no longer needed to satisfy backup retention policies

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Managing Backups: Reporting

Use the RMAN **LIST** command to display information about backup sets, proxy copies, and image copies recorded in the repository. Use this command to list:

- Backups and copies that do not have the **AVAILABLE** status in the RMAN repository
- Backups and copies of data files that are available and can possibly be used in a restore operation
- Backup sets and copies that contain a backup of a specified list of data files or specified tablespaces
- Backup sets and copies that contain a backup of any archived logs with a specified name or range
- Backup sets and copies restricted by tag, completion time, recoverability, or device
- Incarnations of a specified database or of all databases known to the repository
- Stored scripts in the recovery catalog

Use the RMAN **REPORT** command to analyze information in the RMAN repository in more detail. The **REPORT NEED BACKUP** command is used to identify all data files that need a backup. The report assumes that the most recent backup would be used in the event of a restore.

Managing Backups: Reporting (continued)

Using the `REPORT OBSOLETE` command, you can identify files that are no longer needed to satisfy backup retention policies. By default, the `REPORT OBSOLETE` command reports which files are obsolete under the currently configured retention policy. You can generate reports of files that are obsolete according to different retention policies by using `REDUNDANCY` or `RECOVERY WINDOW` retention policy options with the `REPORT OBSOLETE` command.

Refer to the *Oracle Database Backup and Recovery Reference* for detailed syntax information.

Managing Backups: Dynamic Performance Views

Query the following dynamic performance views in the target database to obtain information about your backups:

- V\$BACKUP_SET: Backup sets created
- V\$BACKUP_PIECE: Backup pieces that exist
- V\$DATAFILE_COPY: Copies of data files on disk
- V\$BACKUP_FILES: Information about all files created when creating backups

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Managing Backups: Dynamic Performance Views

There are many views that provide backup-related information. The most commonly used ones are shown in the slide.

If you are using a recovery catalog, you can query corresponding views that contain the same information for each target database registered in the recovery catalog database. The corresponding views have the same name, except that the “V\$” is replaced with “RC_”. Also, they are in the schema owned by the recovery catalog owner. For example, the corresponding views in the recovery catalog, showing the information shown in the slide are: RC_BACKUP_SET, RC_BACKUP_PIECE, RC_DATAFILE_COPY, and RC_BACKUP_FILES.

In order to query the RC_BACKUP_FILES view, you must first execute the following in the recovery catalog database:

```
SQL> CALL DBMS_RCVMAN.SETDATABASE(null,null,null,<dbid>);
```

where <dbid> is the database ID of a target database.

Using Enterprise Manager to View Backup Reports

Results		
Input Summary		
Datafile	Control File	SPFile
Files Backed Up 15	Files Backed Up 3	Files Backed Up 3
Distinct Files 5	Distinct Files 1	Distinct Files 2
Distinct Tablespaces 5	Total Size 28.92M	Total Size 0.01K
Total Size 5.26G	Oldest Checkpoint Time Aug 14, 2007 8:04:16 PM	Oldest Modification Time Aug 14, 2007 7:54:20 PM
Oldest Checkpoint Time Aug 14, 2007 9:19:52 PM	Newest Checkpoint Time Aug 14, 2007 9:21:59 PM	Newest Modification Time Aug 14, 2007 9:01:45 PM
Newest Checkpoint Time Aug 14, 2007 9:21:52 PM		

View Backup Report

The following backup jobs are known to the database. The data is retrieved from the database control file.

Search

Status Start Time Type

Results

Total **19** (Completed **15** Failed **4**)

Backup Name	Status	Start Time	Time Taken	Type	Output Devices	Input Size	Output Size	Output Rate (Per Sec)
2007-08-15T00:18:04	COMPLETED	Aug 15, 2007 12:18:10 AM GMT+07:00	00:01:12	DATAFILE FULL	DISK	1.14M	304.00K	4.22K
2007-08-14T19:55:28	COMPLETED	Aug 14, 2007 7:56:20 PM GMT+07:00	01:25:50	DB INCR	DISK	1.86G	1.80G	367.41K
2007-08-14T02:52:20	COMPLETED	Aug 14, 2007 2:52:22 AM GMT+07:00	00:00:09	CONTROLFILE	SBT_TAPE	9.63M	0.00K	0.00K

Manage

[Schedule Backup](#)
[Manage Current Backups](#)
[Backup Reports](#)
[Manage Restore Points](#)
[Perform Recovery](#)
[View and Manage Transactions](#)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Enterprise Manager to View Backup Reports

You can use the Backup Report page to display lists of backup jobs that are known to the database through the information recorded about them in the database control file.

You can customize the jobs that appear in the Results table by using the Search fields at the top of the page. The Results table lists basic information about each backup job, such as the Start Time, the Time Taken, and the Status of the backup job. You can also use the Results table to drill down to individual, detailed backup job reports by using the link in the Backup Name column.

You can drill down to a Summary of Job page of the backup job by clicking the Status of the job in the Results table, where you can view the contents of the output log.

Click the Backup Name link, and you can use the Backup Report page to display detailed information about that backup. The information displayed on this page is derived from the information recorded in the database control file.

The Backup Report page displays result information in the Result section in various categories, such as Input Summary, containing rollup information about the files that were backed up; Output Summary, containing rollup information about the Backup Sets and Image Copies; and then Inputs and Outputs sections that display tables containing detailed job information about the data files, control files, backup sets, backup pieces, and image copies.

Managing Backups: Cross-Checking and Deleting

Use the following RMAN commands to manage your backups:

- **CROSSCHECK:** Verifies the status of backups and copies recorded in the RMAN repository against media such as disk or tape
- **DELETE EXPIRED:** Removes only files whose status in the repository is **EXPIRED**
- **DELETE OBSOLETE:** Deletes backups that are no longer needed

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Managing Backups: Cross-Checking and Deleting

Use the **CROSSCHECK** command to ensure that data about backups in the recovery catalog or control file is synchronized with actual files on disk or in the media management catalog. The **CROSSCHECK** command operates only on files that are recorded in the RMAN repository.

The **CROSSCHECK** command checks only objects marked **AVAILABLE** or **EXPIRED** by examining the files on disk for **DISK** channels or by querying the media manager for **sbt** channels. The **CROSSCHECK** command updates the repository records for any files that it is unable to find to **EXPIRED**. It does not delete any files that it is unable to find.

The **DELETE** command can remove any file that the **LIST** and **CROSSCHECK** commands can operate on. For example, you can delete backup sets, archived redo logs, and data file copies. The **DELETE** command removes both the physical file and the catalog record for the file. The **DELETE OBSOLETE** command deletes backups that are no longer needed. It uses the same **REDUNDANCY** and **RECOVERY WINDOW** options as **REPORT OBSOLETE**.

If you delete backups without using RMAN, you can use the **UNCATALOG** command to remove the files from the recovery catalog, or you can use the **CROSSCHECK** and **DELETE EXPIRED** commands.

Refer to the *Oracle Database Backup and Recovery Reference* for detailed syntax information.

Quiz

A full database backup can be used as the basis for incremental backups.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Quiz

RMAN can always take a backup when the database is closed.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Summary

In this lesson, you should have learned how to:

- Create image file backups
- Create a whole database backup
- Create a full database backup
- Enable fast incremental backup
- Create duplex backup sets
- Back up a backup set
- Create RMAN multi-section backup
- Create an archival backup for long-term retention
- Report on and maintain backups

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 5 Overview: Creating Backups

This practice covers the following topics:

- Taking an archival backup
- Enabling block change tracking
- Recovering from a damaged block
- Reporting on existing backups
- Backing up the control file

ORACLE

Copyright © 2009, Oracle. All rights reserved.

6

Restore and Recovery Tasks

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

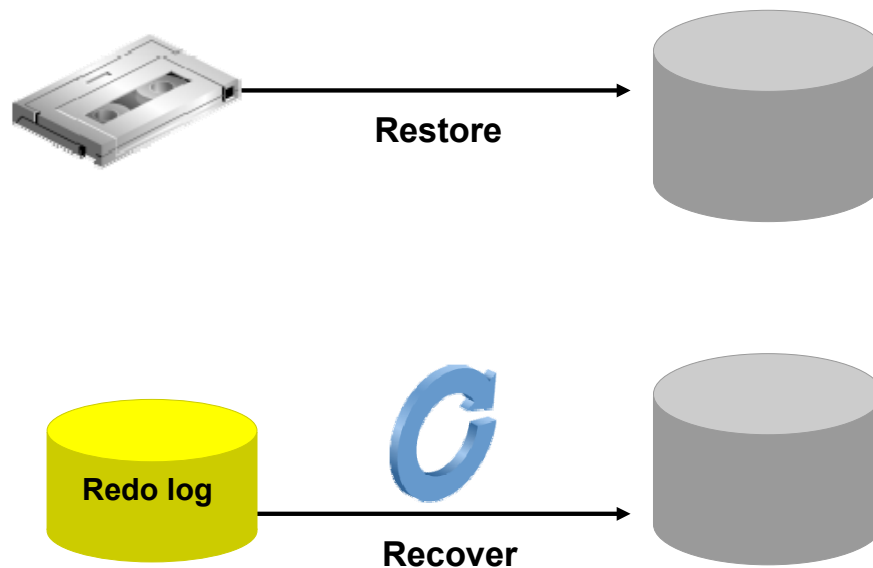
After completing this lesson, you should be able to:

- Describe the causes of file loss and determine the appropriate action
- Describe major recovery operations
- Back up and recover a control file
- Recover from a lost redo log group

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring and Recovering



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring and Recovering

The “recovery” portion of backup and recovery tasks includes two major types of activities: restoring and recovering. *Restoring* a file is the process of copying a backup into place to be used by the database. This is necessary if, for example, a file is damaged because the physical disk it is on fails. This is usually due to hardware problems, such as disk write errors, or controller failure. In that case, a backup of the file needs to be copied onto a new (or repaired) disk.

Recovering the file entails applying redo such that the state of the file is brought forward in time, to whatever point you want. That point is usually as close to the time of failure as possible.

In the database industry, these two operations are often referred to, collectively, with the single term “recovery.”

Causes of File Loss

File loss can be caused by:

- User error
- Application error
- Media failure



ORACLE

Copyright © 2009, Oracle. All rights reserved.

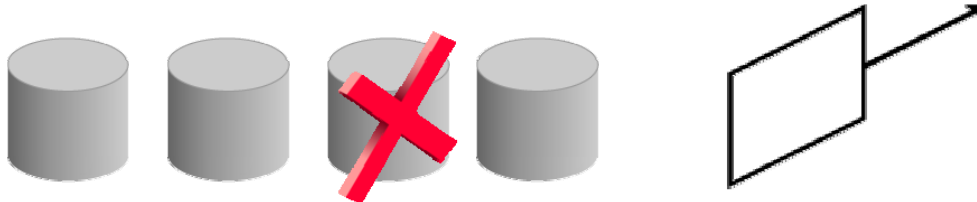
Causes of File Loss

Files can be lost or damaged due to:

- **User error:** An administrator may inadvertently delete or copy over a necessary operating system file.
- **Application error:** An application or script can also have a logic error in it, as it processes database files, resulting in a lost or damaged file.
- **Media failure:** A disk drive or controller may fail fully or partially, and introduce corruption into files, or even cause a total loss of files.

Critical Versus Noncritical

A noncritical file loss is one where the database can continue to function.



You fix the problem by taking one of these actions:

- Create a new file.
- Rebuild the file.
- Recover the lost or damaged file.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Critical Versus Noncritical

A noncritical file is one that the database and most applications can operate without. For example, if the database loses one multiplexed redo log file, there are still other redo log file copies that can be used to keep the database operating.

Although the loss of a noncritical file does not cause the database to crash, it can impair the functioning of the database. For example:

- The loss of an index tablespace can cause applications and queries to run much slower, or even make the application unusable, if the indexes were used to enforce constraints.
- The loss of an online redo log group, as long as it is not the current online log group, can cause database operations to be suspended (when LGWR next tries to write to the group) until new log files are generated.
- The loss of a temporary tablespace can prevent users from running queries or creating indexes until they have been assigned to a new temporary tablespace.

Automatic Tempfile Recovery

SQL statements that require temporary space to execute may fail if one of the tempfiles is missing.

```
SQL> select * from big_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13;
select * from big_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13
*
ERROR at line 1:
ORA-01565: error in identifying file
'/u01/app/oracle/oradata/orcl/temp01.dbf'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
```

Good news:

- Automatic re-creation of temporary files at startup
- (Manual re-creation also possible)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Automatic Tempfile Recovery

If a temporary file (tempfile) belonging to the temporary tablespace is lost or damaged, the extents in that file will not be available. This problem may manifest itself as an error during the execution of SQL statements that require temporary space for sorting.

The SQL statement shown in the slide has a long list of columns to order by, which results in the need for temporary space. The missing file error is encountered when this statement requiring a sort is executed.

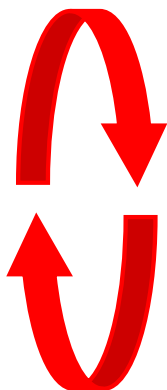
The Oracle database instance can start up with a missing temporary file. If any of the temporary files do not exist when the database instance is started, they are created automatically and the database opens normally. When this happens, a message like the following appears in the alert log during startup:

```
Re-creating tempfile /u01/app/oracle/oradata/orcl/temp01.dbf
```

In the unlikely case that you decide a manual recreation serves you better, use the following commands:

```
SQL> ALTER TABLESPACE temp ADD TEMPFILE
'/u01/app/oracle/oradata/orcl/temp02.dbf' SIZE 20M;
SQL> ALTER TABLESPACE temp DROP TEMPFILE
'/u01/app/oracle/oradata/orcl/temp01.dbf';
```


Log Group Status: Review



A redo log group has a status of one of the following values at any given time:

- **CURRENT:** The LGWR process is currently writing redo data to it.
- **ACTIVE:** It is no longer being written to, but it is still required for instance recovery.
- **INACTIVE:** It is no longer being written to, and it is no longer required for instance recovery.

ORACLE

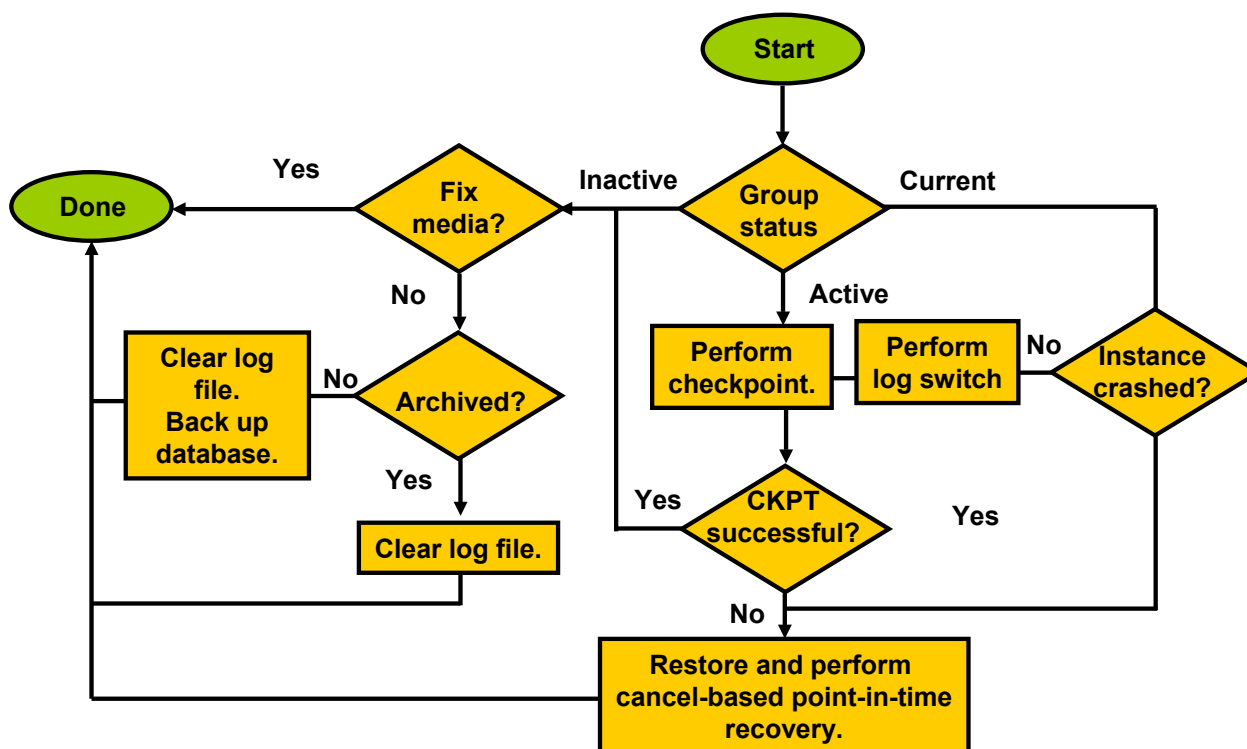
Copyright © 2009, Oracle. All rights reserved.

Log Group Status: Review

To deal with the loss of redo log files, it is important to understand the possible states of redo log groups. Redo log groups cycle through three different states as part of the normal running of the Oracle database. They are, in order of the cycle:

- **CURRENT:** This state means that the redo log group is being written to by LGWR to record redo data for any transactions going on in the database. The log group remains in this state until there is a switch to another log group.
- **ACTIVE:** The redo log group still contains redo data that is required for instance recovery. This is the status during the time when a checkpoint has not yet executed that would write out to the data files all data changes that are represented in the redo log group.
- **INACTIVE:** The checkpoint discussed above has indeed executed, meaning that the redo log group is no longer needed for instance recovery, and is free to become the next CURRENT log group.

Recovering from the Loss of a Redo Log Group



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering from the Loss of a Redo Log Group

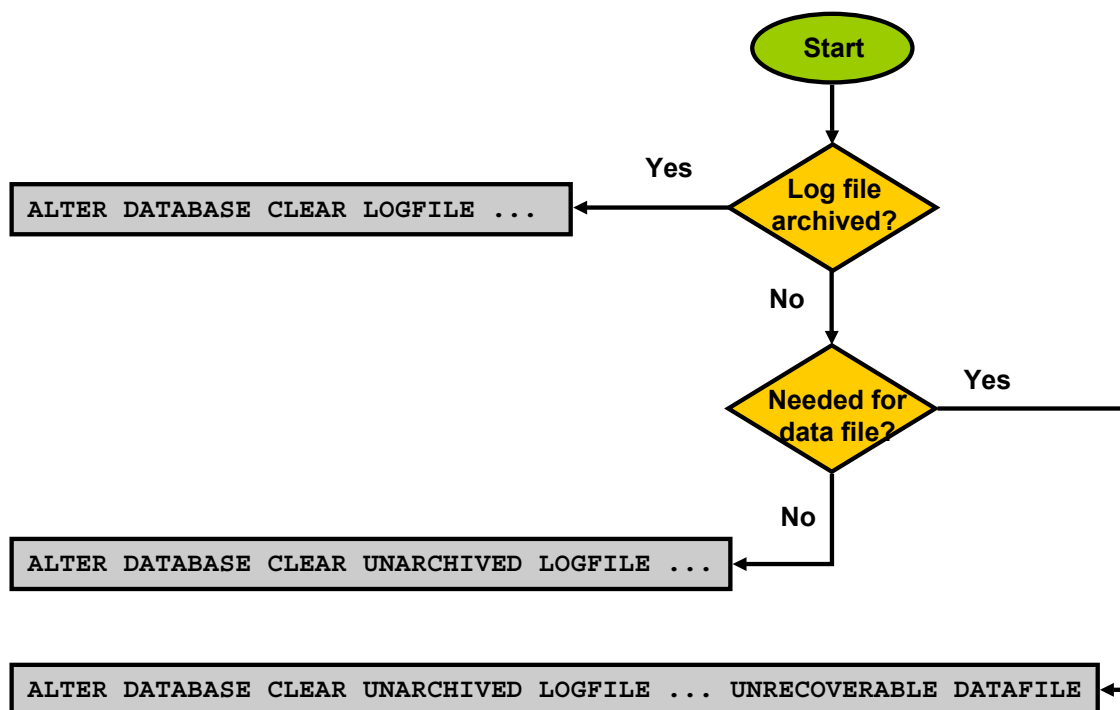
If you have lost an entire redo log group, then all copies of the log files for that group are unusable or gone.

The simplest case is where the redo log group is in the `INACTIVE` state. That means it is not currently being written to, and it is no longer needed for instance recovery. If the problem is temporary, or you are able to fix the media, then the database continues to run normally, and the group is reused when enough log switch events occur. Otherwise, if the media cannot be fixed, you can clear the log file. When you clear a log file, you are indicating that it can be reused.

If the redo log group in question is `ACTIVE`, then, even though it is not currently being written to, it is still needed for instance recovery. If you are able to perform a checkpoint, then the log file group is no longer needed for instance recovery, and you can proceed as if the group were in the inactive state.

If the log group is in the `CURRENT` state, then it is, or was, being actively written to at the time of the loss. You may even see the `LGWR` process fail in this case. If this happens, the instance crashes. Your only option at this point is to restore from backup, perform cancel-based point-in-time recovery, and then open the database with the `RESETLOGS` option.

Clearing a Log File



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Clearing a Log File

Clear a log file using this command:

```
ALTER DATABASE CLEAR [UNARCHIVED] LOGFILE GROUP <n>
[UNRECOVERABLE DATAFILE]
```

When you clear a log file, you are indicating that it can be reused. If the log file has already been archived, the simplest form of the command can be used. Use the following query to determine which log groups have been archived:

```
SQL> SELECT GROUP#, STATUS, ARCHIVED FROM V$LOG;
```

For example, the following command clears redo log group 3, which has already been archived:

```
SQL> ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

If the redo log group has not been archived, then you must specify the UNARCHIVED keyword. This forces you to acknowledge that it is possible that there are backups that rely on that redo log for recovery, and you have decided to forgo that recovery opportunity. This may be satisfactory for you, especially if you take another backup right after you correct the redo log group problem; you then no longer need that redo log file.

It is possible that the redo log is required to recover a data file that is currently offline.

Recovering from a Lost Index Tablespace

- A tablespace that contains only indexes may be recovered without performing a RECOVER task.
- If a data file that belongs to an index-only tablespace is lost, it may be simpler to re-create the tablespace and re-create the indexes.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering from a Lost Index Tablespace

Indexes are computed objects, in that they do not provide any original data, and they are only a different representation of data that already exists. So, in most cases, indexes can be re-created easily. If you have a tablespace that contains only indexes, recovering from a loss of a data file belonging to that tablespace can be simplified.

When a data file like this is lost, you can perform the following steps:

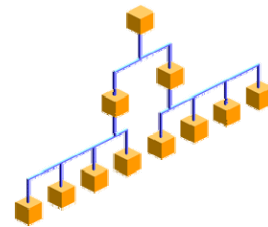
1. Drop the data file.
2. Drop the tablespace.
3. Re-create the index tablespace.
4. Re-create the indexes that were in the tablespace.

Re-Creating Indexes

Use options to reduce the time it takes to re-create the index:

- PARALLEL
- NOLOGGING

```
SQL> CREATE INDEX rname_idx  
2 ON hr.regions (region_name)  
3 PARALLEL 4;
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Re-Creating Indexes

When creating or re-creating an index, you can use the following keywords to reduce the creation time:

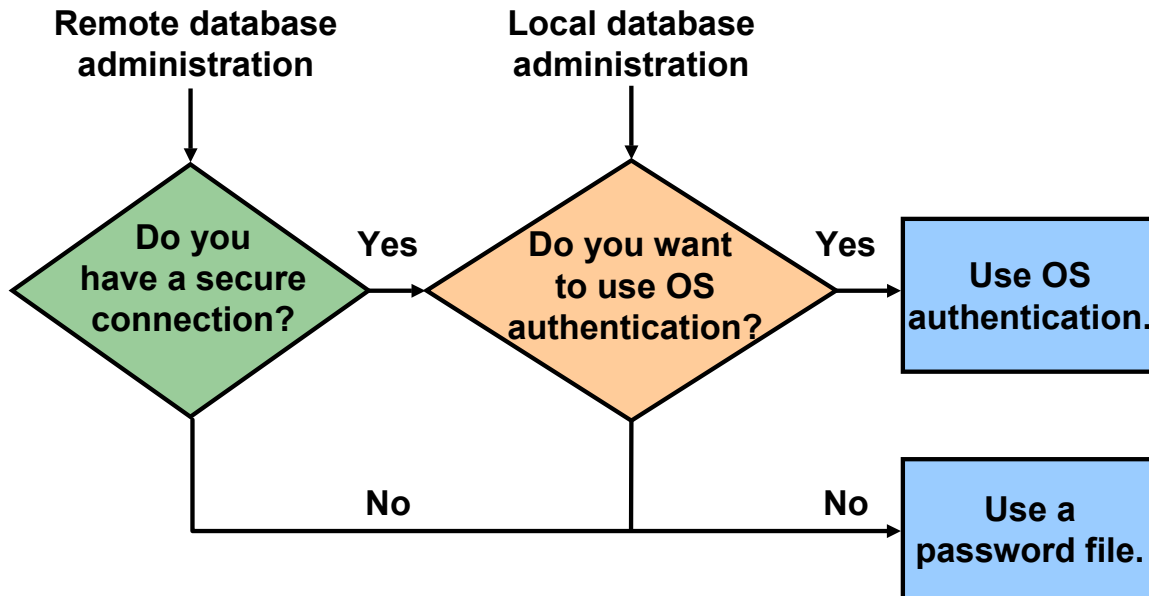
- **PARALLEL (NOPARALLEL is the default):** Multiple processes can work together simultaneously to create an index. By dividing the work necessary to create an index among multiple server processes, the Oracle server can create the index more quickly than if a single server process created the index sequentially. The table is randomly sampled and a set of index keys is found that equally divides the index into the same number of pieces as the specified degree of parallelism. A first set of query processes scans the table, extracts the key-and-row ID pairs, and sends each pair to a process in a second set of query processes based on the key. Each process in the second set sorts the keys and builds an index in the usual fashion. After all index pieces are built, the parallel coordinator concatenates the pieces (which are ordered) to form the final index.
- **NOLOGGING:** Using this keyword makes index creation faster because it creates a very minimal amount of redo log entries as a result of the creation process. This greatly minimized redo generation also applies to direct path inserts and Direct Loader (SQL*Loader) inserts. This is a permanent attribute and thus appears in the data dictionary. It can be updated with the ALTER INDEX NOLOGGING/LOGGING command at any time.

Note: NOLOGGING can be overridden, if you are using Data Guard or FORCE LOGGING at the database or tablespace level.

Re-Creating Indexes (continued)

When an index is lost, it may be faster and simpler just to re-create it rather than attempt to recover it. You can use Data Pump Export with the `CONTENT=METADATA_ONLY` parameter to create a dump file containing the SQL commands to re-create the index. You can also use Data Pump Import with the `SQLFILE=<filename>` parameter on a previously created dump file. The Data Pump Export and Import utilities are covered in detail in the *Oracle Database 11g: Administration Workshop I* course. Additional information can be found in *Oracle Database Utilities*.

Authentication Methods for Database Administrators



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Authentication Methods for Database Administrators


Depending on whether you want to administer your database locally on the same machine on which the database resides or to administer many different database servers from a single remote client, you can choose either operating system or password file authentication to authenticate database administrators:

- If the database has a password file and you have been granted the SYSDBA or SYSOPER system privilege, then you can be authenticated by a password file.
- If the server is not using a password file, or if you have not been granted SYSDBA or SYSOPER privileges and are, therefore, not in the password file, you can use operating system authentication. On most operating systems, authentication for database administrators involves placing the operating system username of the database administrator in a special group, generically referred to as OSDBA. Users in that group are granted SYSDBA privileges. A similar group, OSOPER, is used to grant SYSOPER privileges to users.

Operating system authentication takes precedence over password file authentication. Specifically, if you are a member of the OSDBA or OSOPER group for the operating system, and you connect as SYSDBA or SYSOPER, you will be connected with associated administrative privileges *regardless of the username/password that you specify*.

Re-creating a Password Authentication File

```
SQL> grant sysdba to admin2;  
grant sysdba to admin2  
*  
ERROR at line 1:  
ORA-01994: GRANT failed: password file missing or disabled
```



To recover from the loss of a password file:

1. Re-create the password file by using `orapwd`.

```
$ orapwd file=$ORACLE_HOME/dbs/orapworcl password=ora entries=5
```

2. Add users to the password file and assign appropriate privileges to each user.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Re-creating a Password Authentication File

The Oracle database provides a password utility, `orapwd`, to create a password file. When you connect using the `SYSDBA` privilege, you are connecting as the `SYS` schema and not the schema associated with your username. For `SYSOPER`, you are connected to the `PUBLIC` schema. Access to the database using the password file is provided by `GRANT` commands issued by privileged users.

Typically, the password file is not included in backups because, in almost all situations, it can be easily re-created.

It is critically important to the security of your system that you protect your password file and the environment variables that identify the location of the password file. Any user with access to these could potentially compromise the security of the connection.

You should not remove or modify the password file if you have a database or instance mounted using `REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE` or `SHARED`. If you do, you will be unable to reconnect remotely using the password file.

Note: Passwords are case-sensitive, so you must take that into consideration when re-creating the password file. Also if the original password file was created with the `IGNORECASE=Y` option, then it must be recreated with the same option.

Re-creating a Password Authentication File (continued)

Using a Password File

The following are the steps for re-creating the password file:

1. Create the password file by using the password utility `orapwd`.

```
orapwd file=filename password=password entries=max_users
```

where:

- **filename** is the name of the password file (mandatory).
- **password** is the password for SYS (optional). You are prompted for the password if you do not include the password argument.
- **Entries** is the maximum number of distinct users allowed to connect as SYSDBA or SYSOPER. If you exceed this number, you must create a new password file. It is safer to have a larger number. There are no spaces around the “equal to” (=) character.

Example: `orapwd file=$ORACLE_HOME/dbs/orapwU15
password=admin entries=5`

2. Connect to the database by using the password file created in step 1, and grant privileges as needed.

```
SQL> CONNECT sys/admin AS SYSDBA
SQL> grant sysdba to admin2;
```

Password File Locations

UNIX: `$ORACLE_HOME/dbs`

Windows: `%ORACLE_HOME%\database`

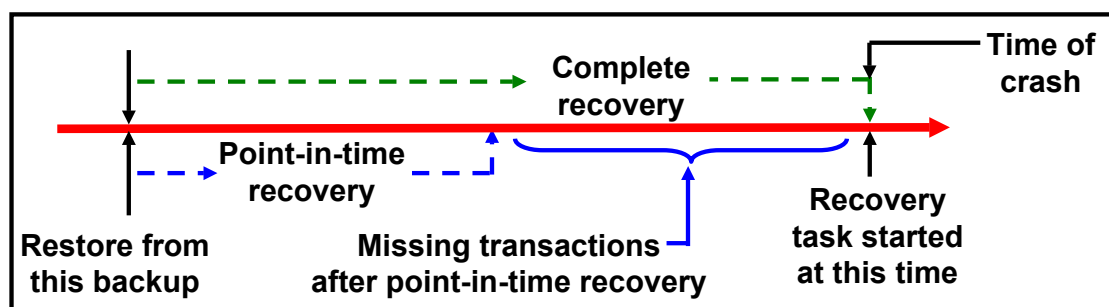
Maintaining the Password File

Delete the existing password file by using operating system commands, and create a new password file by using the password utility.

Comparing Complete and Incomplete Recovery

Recovery can have two kinds of scope:

- Complete recovery: Brings the database up to the present, including all committed data changes made to the point in time when the recovery was requested
- Incomplete or point-in-time recovery: Brings the database up to a specified point in time in the past, before the recovery operation was requested



ORACLE

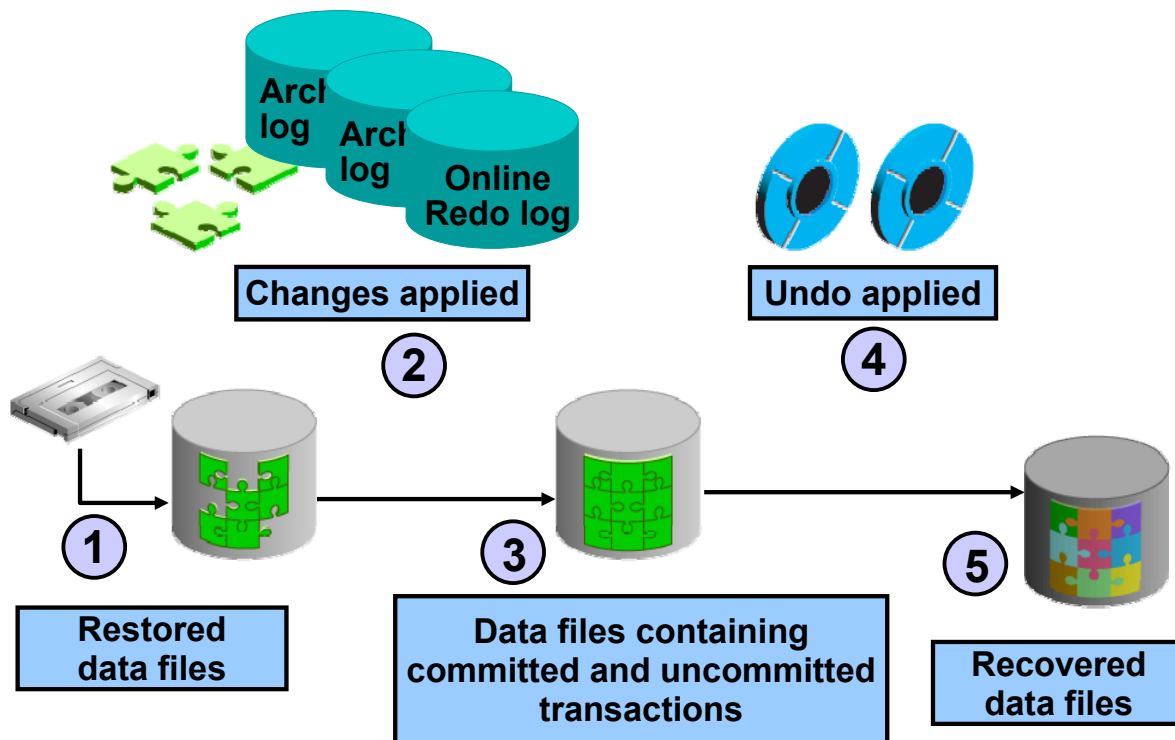
Copyright © 2009, Oracle. All rights reserved.

Comparing Complete and Incomplete Recovery

When you perform complete recovery, you bring the database to the state where it is fully up-to-date, including all committed data modifications to the present time.

Incomplete recovery, however, brings the database to some point in the past point-in-time. It is also known as “Database Point in Time Recovery”. This means there are missing transactions; any data modifications done between the recovery destination time and the present are lost. In many cases, this is the desirable goal because there may have been some changes made to the database that need to be undone. Recovering to a point in the past is a way to remove the unwanted changes.

Complete Recovery Process



ORACLE

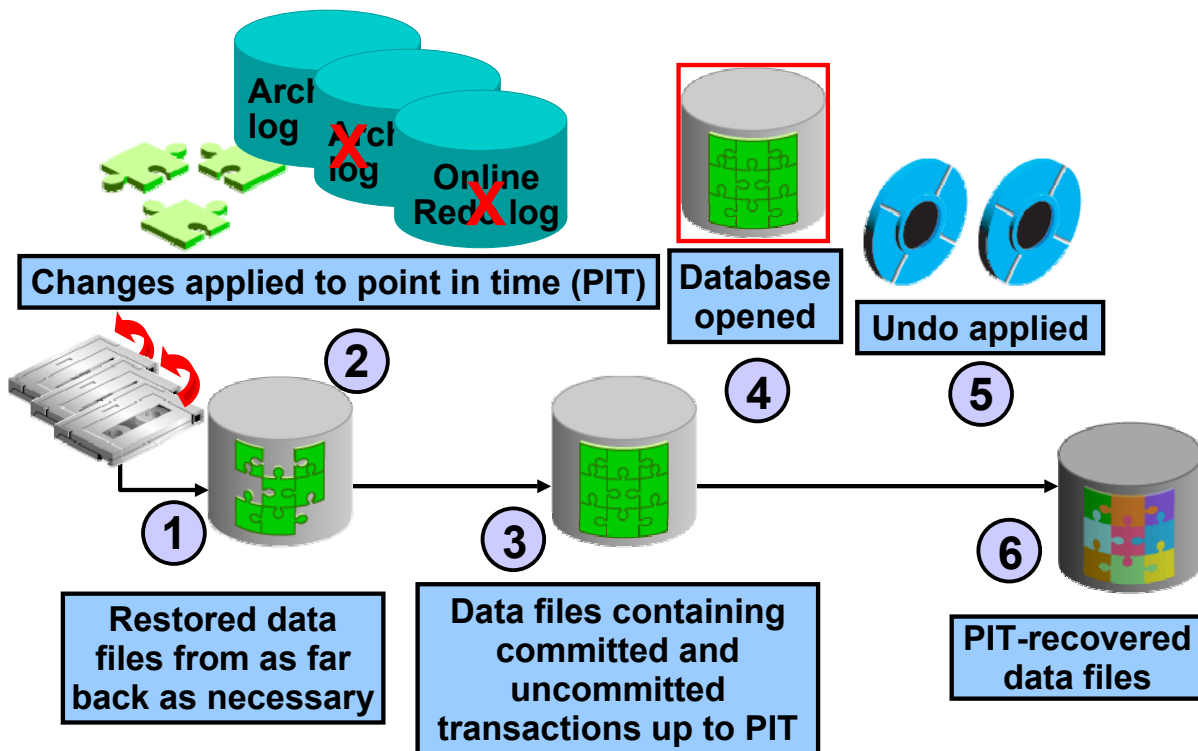
Copyright © 2009, Oracle. All rights reserved.

Complete Recovery Process

The following steps describe what takes place during complete recovery:

1. Damaged or missing files are restored from a backup.
2. Changes from incremental backups, archived redo log files, and online redo log files are applied as necessary. The redo log changes are applied to the data files until the current online log is reached and the most recent transactions have been reentered. Undo blocks are generated during this entire process. This is referred to as rolling forward or cache recovery.
3. The restored data files may now contain committed and uncommitted changes.
4. The undo blocks are used to roll back any uncommitted changes. This is sometimes referred to as transaction recovery.
5. The data files are now in a recovered state and are consistent with the other data files in the database.

Point-in-Time Recovery Process



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Point-in-Time Recovery Process

Incomplete recovery, or database point-in-time recovery, uses a backup to produce a noncurrent version of the database. That is, you do not apply all of the redo records generated after the most recent backup. Perform this type of recovery only when absolutely necessary. To perform Point-in-Time recovery, you need:

- A valid offline or online backup of all the data files made before the recovery point
- All archived logs from the time of the backup until the specified time of recovery

The progression taken to perform an point-in-time recovery is listed below:

- 1. Restore the data files from backup:** The backup that is used may not be the most recent one, if your restore point destination is to be not very recent. This entails either copying files using OS commands or using the RMAN RESTORE command.
- 2. Use the RECOVER command:** Apply redo from the archived redo log files, including as many as necessary to reach the restore point destination.
- 3. State of over-recovery:** Now the data files contain some committed and some uncommitted transactions because the redo can contain uncommitted data.
- 4. Use the ALTER DATABASE OPEN command:** The database is opened before undo is applied. This is to provide higher availability.

Point-in-Time Recovery Process (continued)

5. **Apply undo data:** While the redo was being applied, redo supporting the undo data files was also applied. So the undo is available to be applied to the data files in order to undo any uncommitted transactions. That is done next.
6. **Process complete:** The data files are now recovered to the point in time that you chose.

Point-in-time recovery is the only option if you must perform a recovery and discover that you are missing an archived log containing transactions that occurred sometime between the time of the backup you are restoring from and the target recovery SCN. Without the missing log, you have no record of the updates to your data files during that period. Your only choice is to recover the database from the point in time of the restored backup, as far as the unbroken series of archived logs permits, and then open the database with the RESETLOGS option. All changes in or after the missing redo log file are lost.

Recovering a Read-Only Tablespace

Special user-managed backup and recovery considerations for a read-only tablespace:

- You do not have to put it in backup mode in order to make a copy of its data files.
- You do not have to take the tablespace or data file offline before making a copy of it.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering a Read-Only Tablespace

Because read-only tablespaces are not being written to, there are special considerations to take into account, which can make the recovery process faster and more efficient. You do not have to put a read-only tablespace into backup mode or take it offline before copying it to the backup location. Simply copy it.

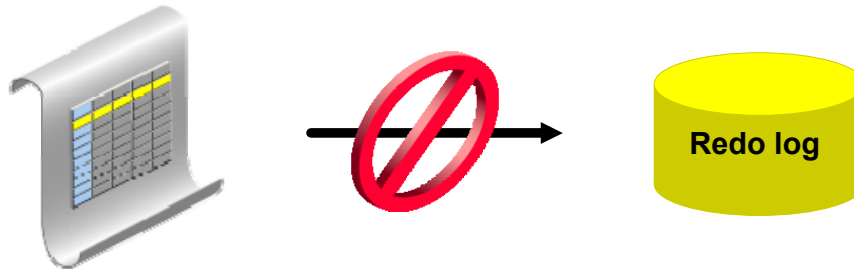
When restoring a read-only tablespace, take the tablespace offline, restore the data files belonging to the tablespace, and then bring the tablespace back online.

Consider the following scenario, where a read-only tablespace is changed to be read/write:

1. Make a backup of a read-only tablespace.
2. Make the tablespace read-write.
3. Recover the tablespace.

The backup you made in step 1 can still be used to recover this tablespace, even though, since the backup was made, the tablespace was made read-write, and has even possibly been written to. In this case, the tablespace requires recovery, after the files are stored from such a backup.

Recovering NOLOGGING Database Objects



```
SQL> CREATE TABLE sales_copy NOLOGGING;  
SQL> INSERT /*+ APPEND */ INTO sales_copy  
2 SELECT * FROM sales_history;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering NOLOGGING Database Objects

Take advantage of the efficiencies of the NOLOGGING attribute of tables and indexes if you can. When you create a table as NOLOGGING, minimal redo data is written to the redo stream to support the creation of the object. This is useful for making large inserts go faster.

In the example in the slide, the SALES_COPY table is created as a NOLOGGING table. As a result, when an insert is done with the APPEND hint, no redo is generated for that particular insert statement. As a result, you cannot recover this transaction on the SALES_HISTORY table. If that is a problem, it is important that you make a backup of whatever tables you populate in this way, right afterward. Then you are able to go to the more recent backup of the table.

If you perform media recovery, and there are NOLOGGING objects involved, they will be marked logically corrupt during the recovery process. In this case, drop the NOLOGGING objects and re-create them.

Use the REPORT UNRECOVERABLE RMAN command to list the names of any tablespaces that contain one or more objects for which a NOLOGGING operation has been performed since the most recent backup of that tablespace.

Recovering from the Loss of All Control File Copies: Overview

	Current	Backup
Available	Restore backup control file, perform complete recovery, OPEN RESETLOGS	Restore backup control file, perform complete recovery, OPEN RESETLOGS
Unavailable	Re-create control file, OPEN RESETLOGS	Restore backup control file, perform point-in-time recovery, OPEN RESETLOGS

Online log status

Data file status

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering from the Loss of All Control File Copies: Overview

Loss of all control files should never happen. **Prevention is better than recovery.** Even though you have copies of the control file stored in different locations, there is still the possibility that you will, at some point, have to recover from losing all those copies. If you have lost all copies of the current control file, and have a backup control file, your course of action depends on the status of the online log files and the data files. The chart in the slide shows what to do in each of the situations shown.

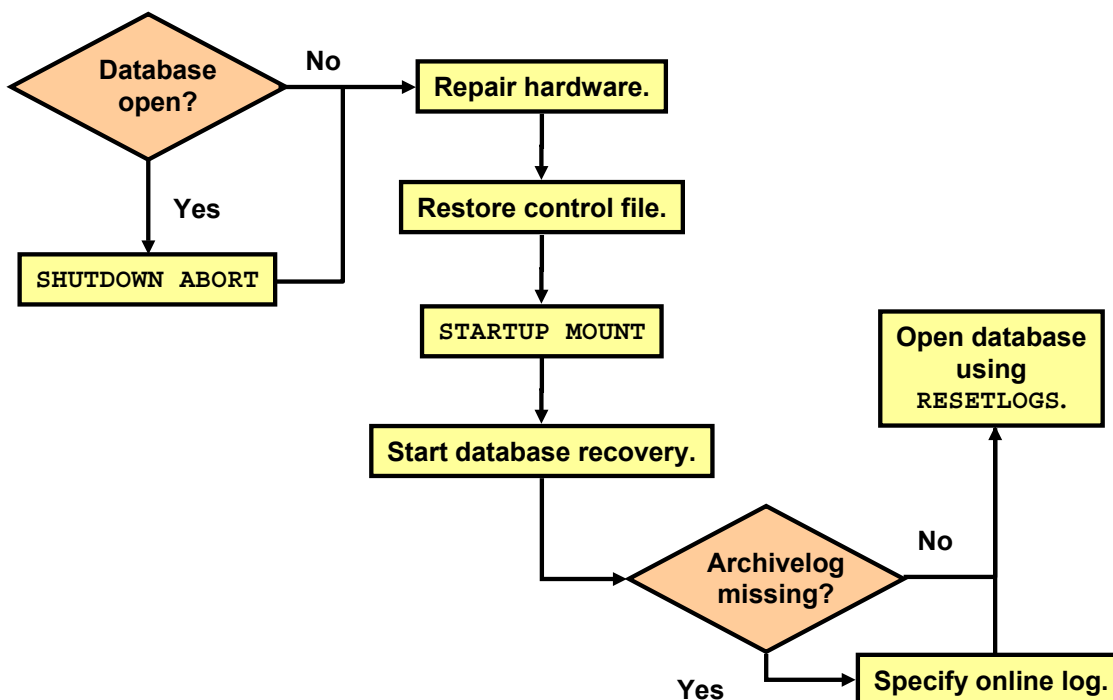
Online Logs Available

If the online logs are available and contain redo necessary for recovery, and the data files are current, then you can restore a backup control file, perform complete recovery, and open the database with the RESETLOGS option. You must specify the file names of the online redo logs during recovery. If the data files are not current, perform the same procedure.

Online Logs Not Available

If the online logs are not available, and the data files are current, then re-create the control file and open RESETLOGS. However, if the data files are not current, restore a backup control file, perform point-in-time recovery, and open RESETLOGS.

Recovering the Control File to the Default Location



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering the Control File to the Default Location

If you need to recover the control file, and the default location is still a valid one, follow the steps shown in the slide. The database must be shut down first. Then repair any hardware, so that the default location is able to remain as a valid one. Restore the control file to the default location. Do this using a command such as this, which copies the backup control file to the default location:

```
% cp /backup/control01.dbf /disk1/oradata/trgt/control01.dbf
% cp /backup/control02.dbf /disk2/oradata/trgt/control02.dbf
```

Mount the database, and start the recovery process. You must specify that a backup control file is being used.

```
SQL> RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL;
```

If, during the recovery process, you are prompted for a missing redo log, it probably means that the missing redo log is an online redo log file. When prompted, supply the name of the online redo log file. After the recovery completes, open the database, specifying the RESETLOGS option.

(More on this topic in the next lesson.)

Quiz

In which of the following cases may the RMAN RECOVER command be issued?

1. The database is in NOARCHIVELOG mode using full backups.
2. The database is in ARCHIVELOG mode using full backups.
3. The database is in NOARCHIVELOG mode using incremental backups.
4. The database is in ARCHIVELOG mode using incremental backups.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 3, 4

Quiz

Your password file is lost, From where can you, as DBA, recover the entries, so that you can re-create your lost password file?

1. Only from the RMAN catalog
2. From the control file
3. From the Enterprise Manager repository
4. From the data dictionary
5. You must manually regrant the SYSOPER, SYSDBA, and SYSASM entries.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 5

Summary

In this lesson, you should have learned how to:

- Describe the causes of file loss and determine the appropriate action
- Describe major recovery operations
- Back up and recover a control file
- Recover from a lost redo log group

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using RMAN to Perform Recovery

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to use RMAN to:

- Perform complete recovery when a critical or noncritical data file is lost
- Recover using incrementally updated backups
- Switch to image copies for fast recovery
- Restore a database onto a new host
- Recover using a backup control file

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

For more details, see the *Oracle Database Backup and Recovery User's Guide*.

Using RMAN RESTORE and RECOVER Commands

- RESTORE command: Restores database files from backup
- RECOVER command: Recovers restored files by applying changes recorded in incremental backups and redo log files

```
RMAN> SQL 'ALTER TABLESPACE inv_tbs OFFLINE IMMEDIATE';
RMAN> RESTORE TABLESPACE inv_tbs;
RMAN> RECOVER TABLESPACE inv_tbs;
RMAN> SQL 'ALTER TABLESPACE inv_tbs ONLINE';
```

- The Enterprise Manager Recovery Wizard creates and runs an RMAN script to perform the recovery.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using RMAN RESTORE and RECOVER Commands

Reconstructing the contents of an entire database or a part of it from a backup typically involves two phases: retrieving a copy of the data file from a backup, and reapplying changes to the file since the backup from the archived and online redo logs, to bring the database to the desired SCN (usually the most recent one).

- RESTORE {DATABASE | TABLESPACE name [,name]... | DATAFILE name [,name] }...

The RESTORE command retrieves the data file onto disk from a backup location on tape, disk, or other media, and makes it available to the database server. RMAN restores from backup any archived redo logs required during the recovery operation. If backups are stored on a media manager, channels must be configured or allocated for use in accessing backups stored there.

- RECOVER {DATABASE | TABLESPACE name [,name]... | DATAFILE name [,name] }...

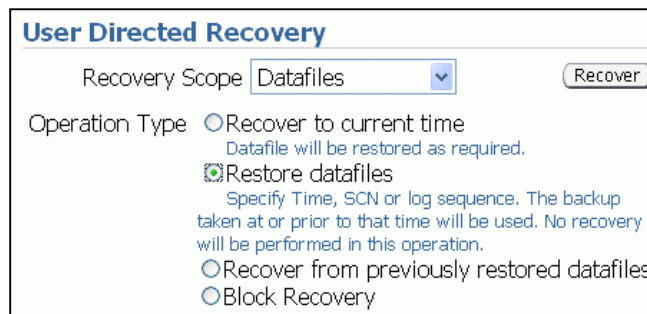
The RECOVER command takes the restored copy of the data file and applies to it the changes recorded in the incremental backups and database's redo logs.

You can also perform complete or point-in-time recovery by using the Recovery Wizard available through Enterprise Manager. On the Availability page, click Perform Recovery in the Backup/Recovery section.

Note: An automated method of detecting the need for recovery, and carrying out that recovery makes use of the Data Recovery Advisor, which is covered in the lesson titled “Diagnosing the Database.”

Performing Complete Recovery: Loss of a Noncritical Data File in ARCHIVELOG Mode

If a data file is lost or corrupted, and that file does not belong to the `SYSTEM` or `UNDO` tablespace, then restore and recover the missing data file.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Complete Recovery: Loss of a Noncritical Data File in ARCHIVELOG Mode

With the database in `ARCHIVELOG` mode, the loss of any data file not belonging to the `SYSTEM` or `UNDO` tablespaces affects only those objects that are in the missing file.

To restore and recover the missing data file using Enterprise Manager, perform the following steps:

1. Click Perform Recovery on the Availability properties page.
2. Select “Datafiles” as Recovery Scope and “Restore datafiles” as Operation Type.
3. Add all data files that need recovery.
4. Specify from what backup the files are to be restored.
5. Determine whether you want to restore the files to the default location or (if a disk or controller is missing) to a new location.
6. Submit the RMAN job to restore and recover the missing files.

Because the database is in `ARCHIVELOG` mode, recovery up to the time of the last commit is possible and users are not required to reenter any data.

Performing Complete Recovery: Loss of a System-Critical Data File in ARCHIVELOG Mode

If a data file is lost or corrupted, and that file belongs to the `SYSTEM`, `UNDO` (or `SYSAUX`) tablespace, then perform the following steps:

1. The instance may or may not shut down automatically. If it does not, use `SHUTDOWN ABORT` to shut the instance down.
2. Mount the database.
3. Restore and recover the missing data file.
4. Open the database.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Complete Recovery: Loss of a System-Critical Data File in ARCHIVELOG Mode

Data files belonging to the `SYSTEM` tablespace or containing `UNDO` data are considered system critical. If Enterprise Manager is used for recovery, the `SYSAUX` tablespace is critical as well. A loss of one of these files requires the database to be restored from the `MOUNT` state (unlike other data files that may be restored with the database open).

Perform the following steps for complete recovery:

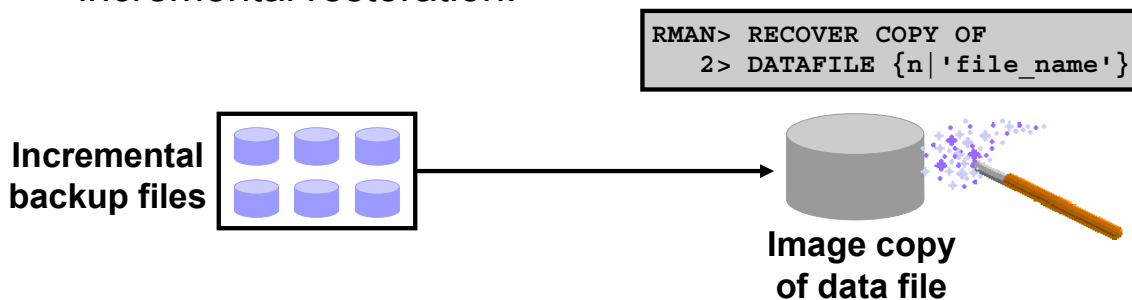
1. If the instance is not already shut down, shut it down.
2. Mount the database.
3. Click Perform Recovery on the Maintenance properties page.
4. Select “Datafiles” as the recovery type, and then select “Restore to current time.”
5. Add all data files that need recovery.
6. Determine whether you want to restore the files to the default location or (if a disk or controller is missing) to a new location.
7. Submit the RMAN job to restore and recover the missing files.
8. Open the database. Users are not required to reenter data because the recovery is up to the time of the last commit.

Note: This kind of recovery situation is detected by the Data Recovery Advisor, which is covered in the lesson titled “Diagnosing the Database.”

Recovering Image Copies

RMAN can recover image copies by using incremental backups:

- Image copies are updated with all changes up to the incremental backup SCN.
- Incremental backup reduces the time required for media recovery.
- There is no need to perform an image copy after the incremental restoration.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering Image Copies

You can use RMAN to apply incremental backups to data file image copies. With this recovery method, you use RMAN to recover a copy of a data file—that is, you roll forward (recover) the image copy to the specified point in time by applying the incremental backups to the image copy. The image copy is updated with all changes up through the SCN at which the incremental backup was taken. RMAN uses the resulting updated data file 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. The following are the benefits of applying incremental backups to data file image copies:

- You reduce the time required for media recovery (using archive logs) because you need to apply archive logs only since the last incremental backup.
- You do not need to perform a full image copy after the incremental restoration.

If the recovery process fails during the application of the incremental backup file, you simply restart the recovery process. RMAN automatically determines the required incremental backup files to apply, from before the image data file copy until the time at which you want to stop the recovery process. If there is more than one version of an image copy recorded in the RMAN catalog, RMAN automatically uses the latest version of the image copy. RMAN reports an error if it cannot merge an incremental backup file with an image copy.

Recovering Image Copies: Example

If you run these commands daily:

```
RMAN> recover copy of database with tag 'daily_inc';  
RMAN> backup incremental level 1 for recover of copy  
2> with tag 'daily_inc' database;
```

This is the result:

	RECOVER	BACKUP
Day 1	Nothing	Create image copies
Day 2	Nothing	Create incremental level 1
Day 3 and onward	Recover copies based on incremental	Create incremental level 1

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovery Image Copies: Example

If you run the commands shown in the slide daily, you get continuously updated image copies of all the database data files at any time.

The chart shows what happens for each run. Note that this algorithm requires some priming; the strategy does not come to fruition until after day 3.

Day 1: The RECOVER command does nothing. There exist no image copies to recover yet. The BACKUP command creates the image copies.

Day 2: The RECOVER command, again, does nothing. This is because there is no incremental backup yet. The BACKUP command creates the incremental backup, now that baseline image copies have been created on day 1.

Day 3: The RECOVER command applies the changes from the incremental backup to the image copies. The BACKUP command takes another incremental backup, which will be used to recover the image copies on day 4. The cycle continues like this.

It is important to use tags when implementing this kind of backup strategy. They serve to link these particular incremental backups to the image copies that are made. Without the tag, the most recent, and possibly the incorrect, incremental backup would be used to recover the image copies.

Performing a Fast Switch to Image Copies

Perform fast recovery by performing the following steps:

1. Take data files offline.
2. Use the `SWITCH TO ... COPY` command to switch to image copies.
3. Recover data files.
4. Bring data files online.

Now the data files are recovered and usable in their new location.

Optionally, do the following to put the files back into their original location:

5. Create an image copy of the data file in the original location.
6. Take data files offline.
7. `SWITCH TO ... COPY`
8. Recover data files.
9. Bring data files online.

```
SQL> SWITCH DATAFILE 'filename' TO COPY;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing a Fast Switch to Image Copies

You can use image copies of data files for fast recovery by performing the following steps:

1. Take the data file offline.
2. Use the `SWITCH TO ... COPY` command to point to the image copy of the files.
3. Recover the data files.
4. Bring the data files online.

At this point, the database is usable, and the data files are recovered. But, if you want to put the data files back into their original location, proceed with the following steps:

5. Create an image copy of the data files in the original location using the `BACKUP AS COPY` command.
6. Take the data files offline.
7. Switch to the copy you made in step 5 using the `SWITCH TO COPY` command.
8. Recover the data files.
9. Bring the data files online.

You can recover data files, tablespaces, tempfiles, or the entire database with this command. The files being switched to must be image copies.

Using SET NEWNAME for Switching Files

- Use the SET NEWNAME command in a RUN block to restore to a nondefault location.

```
RUN
{ ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev2 DEVICE TYPE sbt;
  SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
  SET NEWNAME FOR DATAFILE '/disk1/oradata/prod/users01.dbf'
    TO '/disk2/users01.dbf';
  RESTORE TABLESPACE users;
  SWITCH DATAFILE ALL;
  RECOVER TABLESPACE users;
  SQL "ALTER TABLESPACE users ONLINE";
}
```

- Instead of individual names, specify a default name format for all files in a database or in a named tablespace.
- The default name is used for DUPLICATE, RESTORE, and SWITCH commands in the RUN block.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using SET NEWNAME for Switching Files

The SET NEWNAME command can be used only inside a RUN block. It prepares a name mapping for subsequent operations. In the example in the slide, the SET NEWNAME command defines the location where a restore operation of that data file will be written. When the RESTORE command executes, the users01.dbf data file is restored to /disk2/users01.dbf. It is written there, but the control file is still not pointing to that location. The SWITCH command causes the control file to be updated with the new location.

A more efficient way is to use the SET NEWNAME clause to specify the default name format for all data files in a named tablespace and all data files in the database (rather than setting file names individually, as in database versions prior to Oracle Database 11gR2 (11.2)).

The order of precedence for the SET NEWNAME command is as follows:

1. SET NEWNAME FOR DATAFILE and SET NEWNAME FOR TEMPFILE
2. SET NEWNAME FOR TABLESPACE
3. SET NEWNAME FOR DATABASE

Substitution Variables for SET NEWNAME

Syntax Element	Description
%b	Specifies the file name without the directory path <i>*NEW*</i>
%f	Specifies the absolute file number of the data file for which the new name is generated
%I	Specifies the DBID
%N	Specifies the tablespace name
%U	Specifies a system-generated file name of the format: data-D-%d_id-%I_TS-%N_FNO-%f

RUN

```
{ SET NEWNAME FOR DATAFILE 1 TO '/oradata1/system01.dbf';
SET NEWNAME FOR DATAFILE 2 TO '/oradata2/sysaux01.dbf';
SET NEWNAME FOR DATAFILE 3 TO '/oradata3/undotbs01.dbf';
SET NEWNAME FOR DATAFILE 4 TO '/oradata4/users01.dbf';
SET NEWNAME FOR TABLESPACE example TO '/oradata5/%b';
DUPLICATE TARGET DATABASE TO dupl db; }
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Substitution Variables for SET NEWNAME

To avoid possible name collisions when restoring to another location, use the substitution variables of the SET NEWNAME command. Specify at least one of the following substitution variables: %b, %f, and %U. %I and %N are optional variables.

The example shows the SET NEWNAME FOR TABLESPACE command to set default names with a substitution variable, together with explicit SET NEWNAME clauses.

Performing Restore and Recovery of a Database in NOARCHIVELOG Mode

- If the database is in NOARCHIVELOG mode, and any data file is lost, perform the following tasks:
 - Shut down the instance if it is not already down.
 - Restore the entire database, including all data and control files, from the backup.
 - Open the database.
- Users must reenter all changes made since the last backup.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Restore and Recovery of a Database in NOARCHIVELOG Mode

The loss of any data file from a database in NOARCHIVELOG mode requires complete restoration of the database, including control files and all data files. If you have incremental backups, then you need to perform the restore and recover operations. If the lost data file belongs to a read-only tablespace, you need to restore only that file.

With the database in NOARCHIVELOG mode, recovery is possible only up to the time of the last backup. So users must reenter all changes made since that backup.

For this type of recovery, use the RESTORE and RECOVER commands, or perform the following tasks in Enterprise Manager:

1. Shut down the instance if it is not already down.
2. Click Perform Recovery on the Maintenance properties page.
3. Select Whole Database as the type of recovery.

Using Restore Points

A restore point provides a name to a point in time:

- Now:

```
SQL> CREATE RESTORE POINT before_mods;
```

- Some time in the past:

```
SQL> CREATE RESTORE POINT end_q1 AS OF SCN 100;
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Restore Points

You can give a name to a particular point in time, or an SCN number. This is useful for future reference, when performing point-in-time recovery or flashback operations.

- The first example in the slide creates a restore point that represents the present point in time. If you were about to apply an update of an application or data in the database, and you wanted to refer back to this state of the database, you could use the `BEFORE_MODS` restore point.
- The second example in the slide creates a restore point representing a past SCN, 100. This restore point can be used in the same ways as the previous one.

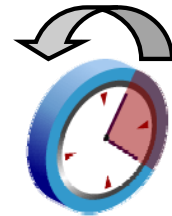
Normally, restore points are maintained in the database for at least as long as specified by the `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter. However, you can use the `PRESERVE` option when creating a restore point, which causes the restore point to be saved until you explicitly delete it.

You can see restore points in the `V$RESTORE_POINT` view with name, SCN, timestamp and other information.

Performing Point-in-Time Recovery

Perform server-managed point-in-time recovery by doing the following:

1. Determine the target point of the restore: SCN, time, restore point, or log sequence number.
2. Set the NLS environment variables appropriately.
3. Mount the database.
4. Prepare and run a RUN block, using the SET UNTIL, RESTORE, and RECOVER commands.
5. Open the database in READONLY mode, and verify that the recovery point is what you wanted.
6. Open the database using RESETLOGS.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Point-in-Time Recovery

You can perform server-managed point-in-time recovery using the following steps. The database must be in ARCHIVELOG mode.

1. Determine the restore target. This can be in terms of a date and time, an SCN, restore point, or log sequence number. For example, if you know that some bad transactions were submitted at 3:00 PM yesterday, then you can choose 2:59 PM yesterday as the target restore point time.
2. Set the National Language Support (NLS) OS environment variables, so that the time constants you provide to RMAN are formatted correctly. These are some example settings:

```
$ export NLS_LANG = american_america.us7ascii  
$ export NLS_DATE_FORMAT = "yyyy-mm-dd:hh24:mi:ss"
```
3. Mount the database. If it is open, you have to shut it down first, as in this example:

```
RMAN> shutdown immediate  
RMAN> startup mount
```

Performing Point-in-Time Recovery (continued)

4. Create a RUN block and run it. The RECOVER and RESTORE commands should be in the same RUN block so that the UNTIL setting applies to both. For example, if you choose to recover to a particular SCN, the RESTORE command needs to know that value so it restores files from backups that are sufficiently old—that is, backups that are from before that SCN. Here is an example of a RUN block:

```
RUN
{
  SET UNTIL TIME '2007-08-14:21:59:00';
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

5. As soon as you open the database for read/write, you have committed to the restore you just performed. So, first, open the database READ ONLY, and view some data, to check whether the recovery did what you expected.

```
RMAN> SQL 'ALTER DATABASE OPEN READ ONLY';
```

6. If satisfied with the results of the recovery, open the database with the RESETLOGS option, as shown:

```
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

Performing Recovery with a Backup Control File

- Restore and mount a backup control file when all copies of the current control file are lost or damaged.
- Execute the `RECOVER` command after restoring the backup control file.
- Open the database with the `RESETLOGS` option after performing complete or point-in-time recovery.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Recovery with a Backup Control File

If you have lost all copies of the current control file, you must restore and mount a backup control file before performing recovery. Your recovery operation may be to recover lost data files or it may be to simply recover the control file. If you are using a recovery catalog, the process is identical to recovery with a current control file because RMAN can use the recovery catalog to obtain RMAN metadata.

Recovery from Loss of Server Parameter File

The FROM MEMORY clause allows the creation of current systemwide parameter settings.

```
SQL> CREATE PFILE [= 'pfile_name' ]  
      FROM { { SPFILE [= 'spfile_name'] } | MEMORY } ;
```

```
SQL> CREATE SPFILE [= 'spfile_name' ]  
      FROM { { PFILE [= 'pfile_name' ] } | MEMORY } ;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

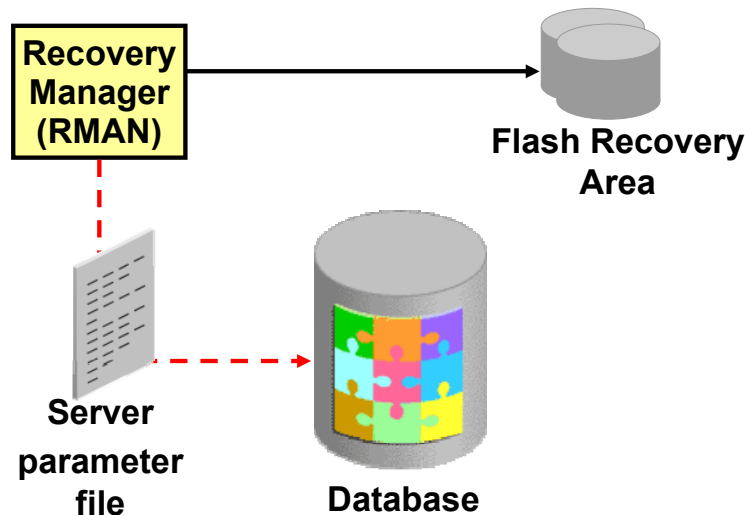
Recovery from Loss of Server Parameter File

The easiest way to recover a server parameter file is to use the FROM MEMORY clause, which creates a text initialization parameter file (PFILE) or server parameter file (SPFILE) using the current systemwide parameter settings. In a RAC environment, the created file contains the parameter settings from each instance.

During instance startup, all parameter settings are logged to the alert.log file. As of Oracle Database 11g, the alert.log parameter dump text is written in valid parameter syntax. This facilitates cutting and pasting of parameters into a separate file, and then using as a PFILE for a subsequent instance. The name of the PFILE or SPFILE is written to the alert.log at instance startup time. In cases when an unknown client-side PFILE is used, the alert log indicates this as well. To support this additional functionality, the COMPATIBLE initialization parameter must be set to 11.0.0.0 or higher.

Restoring the Server Parameter File from the Control File Autobackup

```
RMAN> STARTUP FORCE NOMOUNT;  
RMAN> RESTORE SPFILE FROM AUTOBACKUP;  
RMAN> STARTUP FORCE;
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring the Server Parameter File from the Control File Autobackup

If you have lost the server parameter file and you cannot use the FROM MEMORY clause, then you can restore it from the autobackup. The procedure is similar to restoring the control file from autobackup. If the autobackup is not in the flash recovery area, set the DBID for your database. Issue the RESTORE SPFILE FROM AUTOBACKUP command.

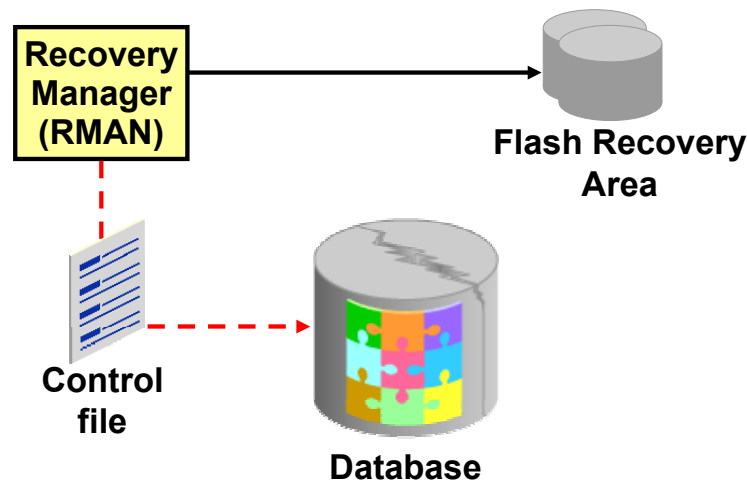
If you are restoring the SPFILE to a nondefault location, specify the command as follows:

```
RESTORE SPFILE TO <file_name> FROM AUTOBACKUP
```

If you are restoring the server parameter file from the Flash Recovery Area, specify the command as follows:

```
RMAN> run {  
2> restore spfile from autobackup  
3> recovery area = '<flash recovery area destination>'  
4> db_name = '<db_name>';  
5> }
```

Restoring the Control File from Autobackup



```
RMAN> STARTUP NOMOUNT;  
RMAN> RESTORE CONTROLFILE FROM AUTOBACKUP;  
RMAN> ALTER DATABASE MOUNT;  
RMAN> RECOVER DATABASE;  
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring the Control File from Autobackup

If you are not using a recovery catalog, you should have autobackup of the control file configured, so that you are able to quickly restore the control file if needed. The commands used for restoring your control file are the same, whether or not you are using a Flash Recovery Area. However, if you are using a Flash Recovery Area, RMAN implicitly cross-checks backups and image copies listed in the control file, and catalogs any files in the Flash Recovery Area not recorded in the restored control file, improving the usefulness of the restored control file in the restoration of the rest of your database.

Use the commands shown in the slide to recover from lost control files. First, start the instance in NOMOUNT mode. It cannot be mounted because there is no control file. Restore the control file from backup. Now that there is a control file, you can mount the database. You must now recover the database, because you now have a backup control file that contains information about an older version of the database. After recovering the database, you can open it. You must specify RESETLOGS because the new control file represents a different instantiation of the database.

Note: Tape backups are not automatically cross-checked after the restoration of a control file. If you are using tape backups, then after restoring the control file and mounting the database, you must cross-check the backups on tape.

Restoring the Control File from Autobackup (continued)

To restore the control file from an autobackup, the database must be in a NOMOUNT state. If the autobackup is not in the flash recovery area, you must set the database identifier (DBID) before issuing the `RESTORE CONTROLFILE FROM AUTOBACKUP` command, as shown in the following example:

```
RMAN> SHUTDOWN ABORT;  
RMAN> STARTUP NOMOUNT;  
RMAN> SET DBID 1090770270;  
RMAN> RESTORE CONTROLFILE FROM AUTOBACKUP;
```

RMAN searches for a control file autobackup. If one is found, RMAN restores the control file from that backup to all the control file locations listed in the `CONTROL_FILES` initialization parameter.

If you have a recovery catalog, you do not have to set the DBID or use the control file autobackup to restore the control file. You can use the `RESTORE CONTROLFILE` command with no arguments:

```
RMAN> RESTORE CONTROLFILE;
```

The instance must be in the NOMOUNT state when you perform this operation, and RMAN must be connected to the recovery catalog. The restored control file is written to all locations listed in the `CONTROL_FILES` initialization parameter.

Use the `RESTORE CONTROLFILE... TO <destination>` command to restore the control file to a nondefault location.

If you have also lost the SPFILE for the database and need to restore it from the autobackup, the procedure is similar to restoring the control file from autobackup. You must first set the DBID for your database, and then use the `RESTORE SPFILE FROM AUTOBACKUP` command.

After you have started the instance with the restored server parameter file, RMAN can restore the control file from the autobackup. After you restore and mount the control file, you have the backup information necessary to restore and recover the database.

After restoring the control files of your database from backup, you must perform complete media recovery and then open your database with the `RESETLOGS` option.

Using Incremental Backups to Recover a Database in NOARCHIVELOG Mode

Use incremental backups to perform limited recovery of a database in NOARCHIVELOG mode.

```
STARTUP FORCE NOMOUNT;  
RESTORE CONTROLFILE;  
ALTER DATABASE MOUNT;  
RESTORE DATABASE;  
RECOVER DATABASE NOREDO;  
ALTER DATABASE OPEN RESETLOGS;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Incremental Backups to Recover a Database in NOARCHIVELOG Mode

You can perform limited recovery of a NOARCHIVELOG mode database by using incremental backups. The incremental backups must be consistent backups.

If you have taken incremental backups, RMAN will use your level 0 and level 1 backups to restore and recover the database.

You must specify the NOREDO option on the RECOVER DATABASE command if the online redo log files are lost or if the redo cannot be applied to the incremental backups. If you do not specify the NOREDO option, RMAN searches for the online redo log files after applying the incremental backups. If the online redo log files are not available, RMAN issues an error message.

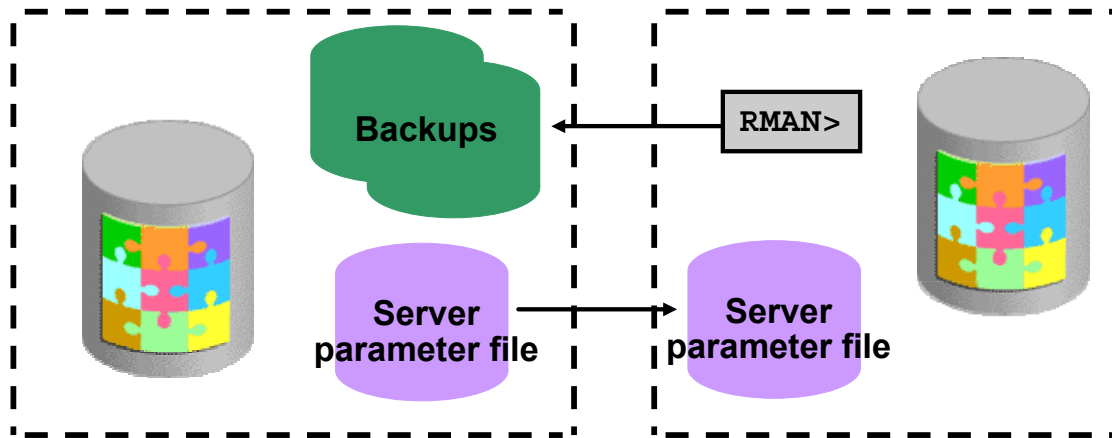
If the current online redo log files contain all changes since the last incremental backup, you can issue the RECOVER DATABASE command without the NOREDO option and the changes will be applied.

Note: You need to restore the control file only if it is not current.

Restoring and Recovering the Database on a New Host

Use the procedure to:

- Perform test restores
- Move a production database to a new host



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring and Recovering the Database on a New Host

Use the procedure described on the following pages to perform test restores. You can also use it to move a production database to a new host.

The database identifier (DBID) for the restored test database is the same as the DBID of the original database. If you are using a recovery catalog and connect to the test database and the recovery catalog database, the recovery catalog is updated with information about the test database. This can impact RMAN's ability to restore and recover the source database.

You should create a duplicate database using the RMAN DUPLICATE command if your goal is to create a new copy of your target database for ongoing use on a new host. The duplicate database is assigned a new DBID that allows it to be registered in the same recovery catalog as the original target database. Refer to the lesson titled "Using RMAN to Duplicate a Database" for detailed information about the DUPLICATE command.

Preparing to Restore the Database to a New Host

To prepare to restore a database, perform the following steps:

- Record the database identifier (DBID) of your source database.
- Copy the source database initialization parameter file to the new host.
- Ensure that source backups, including the control file autobackup, are accessible on the restore host.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Preparing to Restore the Database to a New Host

Perform the steps listed in the slide to prepare for the restore of the database to a new host.

Note: If you are performing a test restore, do not connect to the recovery catalog when restoring the data files. If you connect to the recovery catalog, RMAN records information about the restored data files in the recovery catalog and considers the restored database as the current target database. If your control file is not large enough to contain all of the RMAN repository data on the backups you need to restore and you must use a recovery catalog, then export the catalog and import it into a different schema or database. Use the copied recovery catalog for the test restore.

Restoring the Database to a New Host

Perform the following steps on the restore host to restore the database:

1. Configure the `ORACLE_SID` environment variable.
2. Start RMAN and connect to the target instance in `NOCATALOG` mode.
3. Set the database identifier (DBID).
4. Start the instance in `NOMOUNT` mode.
5. Restore the server parameter file from the backup sets.
6. Shut down the instance.
7. Edit the restored initialization parameter file.
8. Start the instance in `NOMOUNT` mode.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring the Database to a New Host

Perform the steps listed on this page and the next on the restore host to restore the database.

1. Configure the `ORACLE_SID` environment variable as shown in the following example:

```
$ setenv ORACLE_SID orcl
```
2. Start RMAN and connect to the target instance. Do not connect to the recovery catalog as shown in the following example:

```
$ rman TARGET /
```
3. Set the database identifier (DBID). You can find the DBID of your source database by querying the `DBID` column in `V$DATABASE`.

```
RMAN> SET DBID 1090770270;
```
4. Start the instance in `NOMOUNT` mode:

```
RMAN> STARTUP NOMOUNT
```

You will receive an error similar to the following because the server parameter file has not been restored. RMAN uses a “dummy” parameter file to start the instance.

```
startup failed: ORA-01078: failure in processing system
parameters
```

Restoring the Database to a New Host (continued)

5. Restore the server parameter file from the backup sets and shut down the instance as shown in the example:

```
RESTORE SPFILE TO PFILE '?/oradata/test/initiorcl.ora' FROM  
AUTOBACKUP;
```

6. Shut down the instance:

```
SHUTDOWN IMMEDIATE;
```

7. Edit the restored initialization parameter file to change any location-specific parameters, such as those ending in `_DEST`, to reflect the new directory structure.

8. Start the instance in NOMOUNT mode using your edited text initialization parameter file.

```
RMAN> STARTUP NOMOUNT  
      > PFILE='?/oradata/test/initiorcl.ora';
```

Restoring the Database to a New Host

9. Create a RUN block to:
 - Restore the control file
 - Mount the database
10. Create the RMAN recovery script to restore and recover the database.
11. Execute the RMAN script.
12. Open the database with the RESETLOGS option.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring the Database to a New Host (continued)

9. Create a RUN block to restore the control file from an autobackup and mount the database as shown in the example:

```
RUN
{
  RESTORE CONTROLFILE FROM AUTOBACKUP;
  ALTER DATABASE MOUNT;
}
```
10. Query V\$DATAFILE on your new host to determine the database file names as recorded in the control file. Create the RMAN recovery script to restore and recover the database, including the following steps as appropriate:
 - a. Use the SET NEWNAME command to specify the path on your new host for each of the data files that is restored to a different destination than on the original host.
 - b. Use the SQL ALTER DATABASE RENAME FILE command to specify the path for the online redo log files.
 - c. Include the SET UNTIL command to limit recovery to the end of the archived redo log files.
 - d. Include the SWITCH command so that the control file recognizes the new path names as the correct names for the data files.

Restoring the Database to a New Host (continued)

An example of a recovery script follows:

```
RUN
{
SET NEWNAME FOR DATAFILE 1 TO '?/oradata/test/system01.dbf';
SET NEWNAME FOR DATAFILE 2 TO '?/oradata/test/undotbs01.dbf';
SET NEWNAME FOR DATAFILE 3 TO '?/oradata/test/sysaux.dbf';
SET NEWNAME FOR DATAFILE 4 TO '?/oradata/test/users01.dbf';
SET NEWNAME FOR DATAFILE 5 TO '?/oradata/test/example01.dbf';
SQL "ALTER DATABASE RENAME FILE
''/u01/app/oracle/oradata/orcl/redo01.log''
TO ''?/oradata/test/redo01.log'' ";
SQL "ALTER DATABASE RENAME FILE
''/u01/app/oracle/oradata/orcl/redo02.log''
TO ''?/oradata/test/redo02.log'' ";
SQL "ALTER DATABASE RENAME FILE
''/u01/app/oracle/oradata/orcl/redo03.log''
TO ''?/oradata/test/redo03.log'' ";
SET UNTIL SCN 4545727;
RESTORE DATABASE;
SWITCH DATAFILE ALL;
RECOVER DATABASE;
}
```

11. Execute the recovery script.

12. Open the database with the RESETLOGS option:

```
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

After you have completed your test, you can shut down the test database instance and delete the test database with all its files.

Performing Disaster Recovery

- Disaster implies the loss of the entire target database, the recovery catalog database, all current control files, all online redo log files, and all parameter files.
- Disaster recovery includes the restoration and recovery of the target database.
- Minimum required set of backups:
 - Backups of data files
 - Corresponding archived redo logs files
 - At least one control file autobackup

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Disaster Recovery

Disaster recovery includes the restoration and recovery of the target database after the loss of the entire target database, all current control files, all online redo log files, all parameter files, and the recovery catalog database (if applicable).

To perform disaster recovery, the following backups are required as a minimum:

- Backups of data files
- Corresponding archived redo logs generated after the time of the backup
- At least one autobackup of the control file

Note: Refer to the *Oracle Data Guard Concepts and Administration* manual for information about how Oracle Data Guard can provide complete disaster protection.

Performing Disaster Recovery

Basic procedure:

- Restore an autobackup of the server parameter file.
- Start the target database instance.
- Restore the control file from autobackup.
- Mount the database.
- Restore the data files.
- Recover the data files.
- Open the database with the `RESETLOGS` option.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Disaster Recovery (continued)

The basic procedure for performing disaster recovery is outlined in the slide. After you have mounted the database, follow the steps for performing recovery with a backup control file.

Quiz

When you have lost no data files and you recover the backup control file, why is the `RECOVER` command required?

1. To roll forward changes to the control file by resynchronizing from the data files
2. To roll forward changes to the control file by applying redo from the redo logs
3. To roll forward changes to the control file by using the RMAN catalog

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Quiz

With the `RESTORE` command, you restore database files from backup, but you do not apply redo from redo logs.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Summary

In this lesson, you should have learned how to use RMAN to do the following:

- Perform complete recovery when a critical or noncritical data file is lost
- Recover using incrementally updated backups
- Switch to image copies for fast recovery
- Restore a database onto a new host
- Recover using a backup control file

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 7 Overview: Using RMAN to Perform Recovery

This practice covers the following topics:

- Recovering image copies
- Performing fast recovery

ORACLE

Copyright © 2009, Oracle. All rights reserved.

8

Monitoring and Tuning RMAN

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

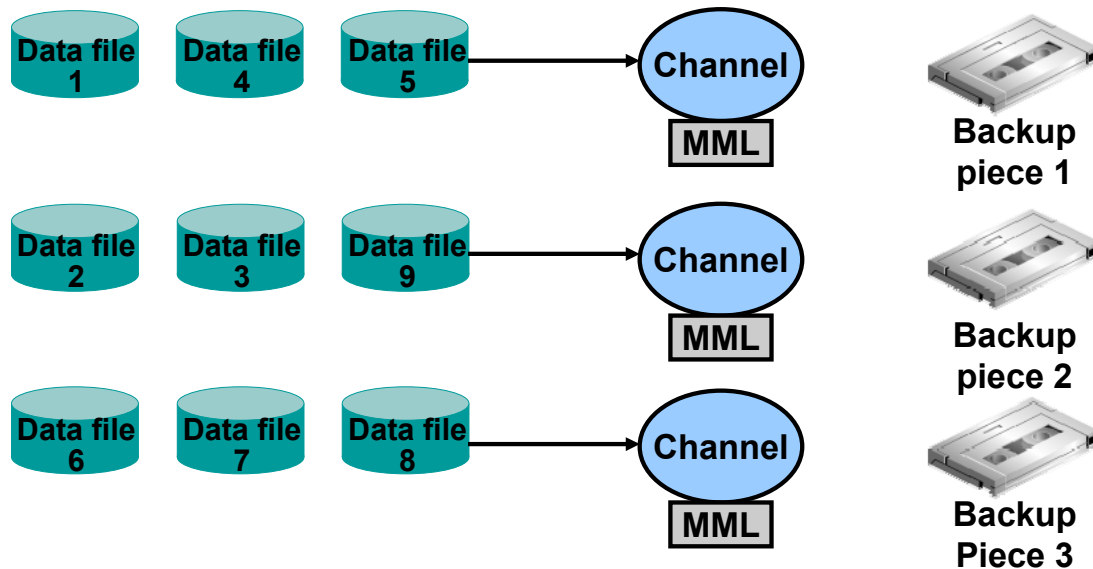
- Monitor the progress of RMAN jobs
- Configure RMAN appropriately for asynchronous I/O
- Configure RMAN multiplexing so as to keep tape drives streaming efficiently
- Evaluate the balance between speed of backup versus speed of recovery
- Explain the effect of the following parameters on RMAN performance: MAXPIECESIZE, FILESPERSET, MAXOPENFILES
- Explain how the RMAN BACKUP DURATION option can cause backups to either execute faster or take longer, (freeing up resources for other processing)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Parallelization of Backup Sets

For performance, allocate multiple channels and assign files to specific channels.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Parallelization of Backup Sets

You can configure parallel backups by setting the `PARALLELISM` option of the `CONFIGURE` command to greater than 1 or by manually allocating multiple channels. RMAN parallelizes its operation and writes multiple backup sets in parallel. The server sessions divide the work of backing up the specified files.

Example

```
RMAN> RUN {  
2>   ALLOCATE CHANNEL c1 DEVICE TYPE sbt;  
3>   ALLOCATE CHANNEL c2 DEVICE TYPE sbt;  
4>   ALLOCATE CHANNEL c3 DEVICE TYPE sbt;  
5>   BACKUP  
6>   INCREMENTAL LEVEL = 0  
7>   (DATAFILE 1,4,5 CHANNEL c1)  
8>   (DATAFILE 2,3,9 CHANNEL c2)  
9>   (DATAFILE 6,7,8 CHANNEL c3);  
10>  SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';  
11> }
```

Parallelization of Backup Sets (continued)

When backing up data files, you can specify the files to be backed up by either their path name or their file number. For example, the following two commands perform the same action:

```
BACKUP DEVICE TYPE sbt DATAFILE '/home/oracle/system01.dbf';  
BACKUP DEVICE TYPE sbt DATAFILE 1;
```

When you create multiple backup sets and allocate multiple channels, RMAN automatically parallelizes its operation and writes multiple backup sets in parallel. The allocated server sessions share the work of backing up the specified data files, control files, and archived redo logs. You cannot stripe a single backup set across multiple channels.

Parallelization of backup sets is achieved by:

- Configuring PARALLELISM to greater than 1 or allocating multiple channels
- Specifying many files to back up

Example

- There are nine files that need to be backed up (data files 1 through 9).
- Assign the data files to a backup set so that each set has approximately the same number of data blocks to back up (for efficiency).
 - Data files 1, 4, and 5 are assigned to backup set 1.
 - Data files 2, 3, and 9 are assigned to backup set 2.
 - Data files 6, 7, and 8 are assigned to backup set 3.

Note: You can also use the FILESPERSET parameter to limit the number of data files that are included in a backup set.

Monitoring RMAN Sessions

- Query V\$SESSION and V\$PROCESS to identify the relationship between server sessions and RMAN channels.
- If you are monitoring multiple sessions, use the SET COMMAND ID command to correlate a process with a channel during a backup.

```
SQL> COLUMN CLIENT_INFO FORMAT a30
SQL> COLUMN SID FORMAT 999
SQL> COLUMN SPID FORMAT 9999
SQL> SELECT s.sid, p.spid, s.client_info
  2 FROM v$process p, v$session s
  3 WHERE p.addr = s.paddr
  4 AND CLIENT_INFO LIKE 'rman%';
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring RMAN Sessions

To identify which server sessions correspond to which RMAN channels, you can query V\$SESSION and V\$PROCESS. The SPID column of V\$PROCESS identifies the operating system ID number for the process or the thread. On UNIX, the SPID column shows the process ID. On Windows, the SPID column shows the thread ID. There are two basic methods for obtaining this information, depending on whether you have multiple RMAN sessions active concurrently. When only one RMAN session is active, execute the following query on the target database while the RMAN job is running:

```
SQL> COLUMN CLIENT_INFO FORMAT a30
SQL> COLUMN SID FORMAT 999
SQL> COLUMN SPID FORMAT 9999
SQL> SELECT s.sid, p.spid, s.client_info
  2 FROM v$process p, v$session s
  3 WHERE p.addr = s.paddr
  4 AND CLIENT_INFO LIKE 'rman%';
SID SPID CLIENT_INFO
-----
15 2714 rman channel=ORA_SBT_TAPE_1
13 2715 rman channel=ORA_SBT_TAPE_2
```

Monitoring RMAN Sessions (continued)

When multiple RMAN sessions are running, it helps to correlate a process with a channel during a backup by using the SET COMMAND ID command as shown below:

1. In each session, set the command ID to a different value and then back up the desired object. For example, enter the following in session 1:

```
RUN
{
  SET COMMAND ID TO 'sess1';
  BACKUP DATABASE;
}
```

Set the command ID to a string such as sess2 in the job running in session 2:

```
RUN
{
  SET COMMAND ID TO 'sess2';
  BACKUP DATABASE;
}
```

2. Start a SQL*Plus session and then query the joined V\$SESSION and V\$PROCESS views while the RMAN job is being executed. For example, enter:

```
SELECT SID, SPID, CLIENT_INFO
FROM V$PROCESS p, V$SESSION s
WHERE p.ADDR = s.PADDR
AND CLIENT_INFO LIKE '%id=sess%';
```

If you run the SET COMMAND ID command in the RMAN job, then the CLIENT_INFO column is displayed in the following format:

id=command_id,rman channel=channel_id

For example, the following shows a sample output:

SID	SPID	CLIENT_INFO
----	-----	-----
11	8358	id=sess1
15	8638	id=sess2
14	8374	id=sess1,rman channel=c1
9	8642	id=sess2,rman channel=c1

Monitoring RMAN Job Progress

Monitor the progress of backup and restore operations by querying V\$SESSION_LONGOPS.

```
SQL> SELECT OPNAME, CONTEXT, SOFAR, TOTALWORK,
 2  ROUND(SOFAR/TOTALWORK*100,2) "%_COMPLETE"
 3  FROM V$SESSION_LONGOPS
 4  WHERE OPNAME LIKE 'RMAN%'
 5  AND OPNAME NOT LIKE '%aggregate%'
 6  AND TOTALWORK != 0
 7  AND SOFAR <> TOTALWORK;
```

SID	SERIAL#	CONTEXT	SOFAR	TOTALWORK	%_COMPLETE
13	75	1	9470	15360	61.65
12	81	1	15871	28160	56.36

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring RMAN Job Progress

Monitor the progress of backups, copies, and restores by querying the V\$SESSION_LONGOPS view. RMAN uses detail and aggregate rows in V\$SESSION_LONGOPS. *Detail rows* describe the files that are being processed by one job step. *Aggregate rows* describe the files that are processed by all job steps in an RMAN command. A job step is the creation or restoration of one backup set or data file copy. The detail rows are updated with every buffer that is read or written during the backup step, so their granularity of update is small. The aggregate rows are updated when each job step is completed, so their granularity of update is large.

Note: Set the STATISTICS_LEVEL parameter to TYPICAL (the default value) or ALL to populate the V\$SESSION_LONGOPS view.

The relevant columns in V\$SESSION_LONGOPS for RMAN include:

- **OPNAME:** A text description of the row. Detail rows include RMAN:datafile copy, RMAN:full datafile backup, and RMAN:full datafile restore.
- **CONTEXT:** For backup output rows, the value of this column is 2. For all the other rows except proxy copy (which does not update this column), the value is 1.

Monitoring RMAN Job Progress (continued)

- **SO FAR:** For image copies, the number of blocks that have been read; for backup input rows, the number of blocks that have been read from the files that are being backed up; for backup output rows, the number of blocks that have been written to the backup piece; for restores, the number of blocks that have been processed to the files that are being restored in this one job step; and for proxy copies, the number of files that have been copied
- **TOTALWORK:** For image copies, the total number of blocks in the file; for backup input rows, the total number of blocks to be read from all files that are processed in this job step; for backup output rows, the value is 0 because RMAN does not know how many blocks it will write into any backup piece; for restores, the total number of blocks in all files restored in this job step; and for proxy copies, the total number of files to be copied in this job step

Interpreting RMAN Message Output

RMAN troubleshooting information can be found in:

- RMAN command output
- RMAN trace file
- Alert log
- Oracle server trace file
- `sbtio.log` file

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Interpreting RMAN Message Output

The RMAN command output contains actions that are relevant to the RMAN job as well as error messages that are generated by RMAN, the server, and the media vendor. RMAN error messages have an `RMAN-nnnn` prefix. The output is displayed to the terminal (standard output) but can be written to a file by defining the `LOG` option or by shell redirection.

The RMAN trace file contains the `DEBUG` output and is used only when the `TRACE` command option is used.

The alert log contains a chronological log of errors, nondefault initialization parameter settings, and administration operations. Because it records values for overwritten control file records, it can be useful for RMAN maintenance when operating without a recovery catalog.

The Oracle trace file contains detailed output that is generated by Oracle server processes. This file is created when an `ORA-600` or `ORA-3113` (following an `ORA-7445`) error message occurs, whenever RMAN cannot allocate a channel, and when the Media Management Library fails to load. It can be found in `USER_DUMP_DEST`.

The `sbtio.log` file contains vendor-specific information that is written by the media management software and can be found in `USER_DUMP_DEST`. Note that this log does not contain Oracle server or RMAN errors.

Using the DEBUG Option

- The `DEBUG` option is used to:
 - View the PL/SQL that is generated
 - Determine precisely where an RMAN command is hanging or faulting
- The `DEBUG` option is specified at the RMAN prompt or within a run block.
- The `DEBUG` option creates an enormous amount of output, so redirect the output to a trace file:

```
$ rman target / catalog rman/rman debug trace trace.log
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the DEBUG Option

The `DEBUG` option displays all SQL statements that are executed during RMAN compilations and the results of these executions. Any information that is generated by the recovery catalog PL/SQL packages is also displayed. In the following example, the `DEBUG` output is written during the backup of data file 3, but not data file 4:

```
RMAN> run {
        debug on;
        allocate channel c1 type disk;
        backup datafile 3;
        debug off;
        backup datafile 4; }
```

Remember that the `DEBUG` output can be voluminous, so make sure that you have adequate disk space for the trace file. This simple backup session that does not generate any errors creates a trace file that is almost half a megabyte in size:

```
$ rman target / catalog rman/rman debug trace sample.log
RMAN> backup database;
RMAN> host "ls -l sample.log";
-rw-r--r--  1 user02   dba             576270 Apr  6 10:38 sample.log
host command complete
```

Interpreting RMAN Error Stacks

- Read the stack from bottom to top.
- Look for Additional information.
- RMAN-03009 identifies the failed command.

```

RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03009: failure of backup command on c1 channel at
              09/04/2001 13:18:19
ORA-19506: failed to create sequential file,
              name="07d36ecp_1_1", parms=""
ORA-27007: failed to open file
SVR4 Error: 2: No such file or directory
Additional information: 7005
Additional information: 1
ORA-19511: Error from media manager layer,error text:

```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Interpreting RMAN Error Stacks

Because of the amount of data that RMAN logs, you may find it difficult to identify the useful messages in the RMAN error stack. Note the following tips and suggestions:

- Because many of the messages in the error stack are not meaningful for troubleshooting, try to identify the one or two errors that are most important.
- Check for a line that says Additional information followed by an integer. This line indicates a media management error. The integer that follows refers to code that is explained in the text of the error message.
- Read the messages from bottom to top because this is the order in which RMAN issues the messages. The last one or two errors that are displayed in the stack are often informative.
- Look for the RMAN-03002 or RMAN-03009 message immediately following the banner. The RMAN-03009 is the same as RMAN-03002 but includes the channel ID. If the failure is related to an RMAN command, then these messages indicate which command failed. The syntax errors generate an RMAN-00558 error.

Tuning RMAN

- RMAN BACKUP and RESTORE operations perform the following tasks:
 - Read or write data.
 - Process data by copying and validating blocks.
- The slowest of these tasks is referred to as a bottleneck, for any particular process.
- Tuning RMAN requires that the bottlenecks be identified and addressed.
- Performance of backup versus recovery operations can be balanced to suit your needs.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Tuning RMAN

RMAN backup and restore operations perform the following distinct tasks:

- Reading or writing input data
- Processing data by validating and copying blocks from the input to the output buffers

The slowest of these tasks is called a *bottleneck*. RMAN tuning involves identifying the bottleneck (or bottlenecks) and attempting to make it more efficient by using RMAN commands, initialization parameter settings, or adjustments to the physical media. The key to tuning RMAN is in understanding the input/output (I/O). The backup and restore jobs of RMAN use two types of I/O buffers: disk and tertiary storage (usually tape). When performing a backup, RMAN reads input files by using disk buffers and writes the output backup file by using either the disk or the tape buffer. When performing restores, RMAN reverses these roles. I/O can be synchronous and asynchronous. Synchronous devices perform only one I/O task at a time. Therefore, you can easily determine how much time the backup jobs require. In contrast to synchronous I/O (SIO), asynchronous I/O (AIO) can perform more than one task at a time. To tune RMAN effectively, you must thoroughly understand the concepts of synchronous and asynchronous I/O, disk and tape buffers, and channel architecture. With an understanding of these concepts, you can use fixed views to monitor bottlenecks.

Tuning RMAN (continued)

You may be able to take advantage of some backup and recovery features that allow you to balance the performance of backup operations versus recovery operations. For example, if you require shorter recovery time, then you may want to perform image copy recovery on a regular basis. That takes more resources to prepare for recovery, but would lessen the amount of time needed to perform the recovery.

RMAN Multiplexing

- For reads:

Multiplexing Level	Allocation Rule
Level <= 4	1 MB buffers are allocated so that the total buffer size for all input files is 16 MB.
4 < Level <= 8	512 KB are allocated so that the total buffer size for all files is less than 16 MB.
Level > 8	RMAN allocates four 128 KB disk buffers per channel for each file, so that the total size is 512 KB per channel for each file.

- For writes, each channel allocates four output buffers of 1 MB each.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

RMAN Multiplexing

RMAN uses two different types of buffers for I/O: disk and tape. RMAN multiplexing determines how RMAN allocates disk buffers. *RMAN multiplexing* is the number of files in a backup read simultaneously and then written to the same backup piece. The degree of multiplexing depends on the `FILESERSET` parameter of the `BACKUP` command as well as the `MAXOPENFILES` parameter of the `CONFIGURE CHANNEL` command or `ALLOCATE CHANNEL` command. Note: RMAN multiplexing is set at the channel level.

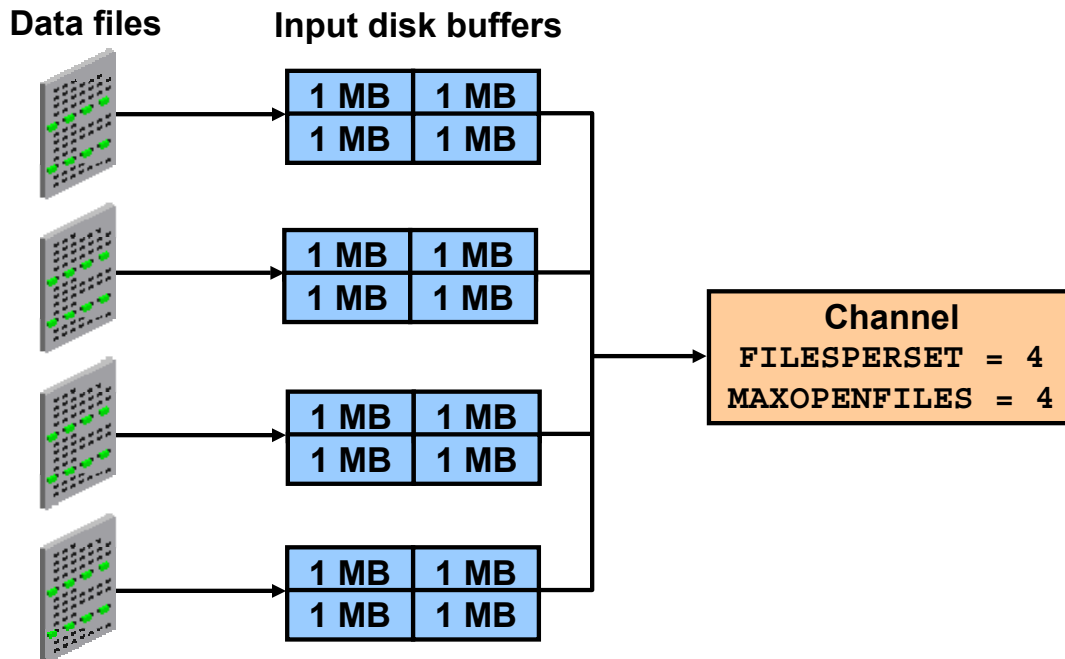
For example, assume that you back up two data files with one channel. You set `FILESERSET` to 3 and `MAXOPENFILES` to 8. In this case, the number of files in each backup set is 2 (the lesser of `FILESERSET` and the files read by each channel) and the level of multiplexing is 2 (the lesser of `MAXOPENFILES` and the number of files in each backup set). When RMAN backs up from disk, it uses the algorithm that is described in the table shown in the slide.

For writing, each channel allocates four output buffers of size 1 MB each.

These buffers are allocated from the PGA unless `DBWR_IO_SLAVES` is set to a nonzero value.

Note: For best recovery performance, do not set `FILESERSET` to a value greater than 8.

Allocating Disk Buffers: Example



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Allocating Disk Buffers: Example

In the example shown in the slide, one channel is backing up four data files. MAXOPENFILES is set to 4 and FILESPERSET is set to 4. The level of multiplexing is 4 in this example. The total size of the buffers for each data file is 4 MB. To calculate the total size of the buffers that are allocated in a backup set, multiply the total bytes for each data file by the number of data files that are being concurrently accessed by the channel, and then multiply this number by the number of channels.

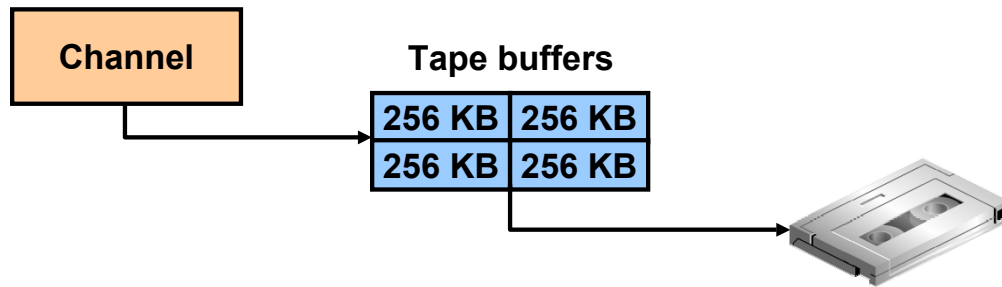
Assume that you use one channel to back up four data files, and use the settings that are shown in the slide. In this case, multiply as follows to obtain the total size of the buffers that are allocated for the backup:

$4 \text{ MB per data file} \times 1 \text{ channel} \times 4 \text{ data files per channel} = 16 \text{ MB}$

Set the MAXOPENFILES parameter so that the number of files that are read simultaneously is just enough to use the output device fully. This consideration is important when the output device is a tape.

Allocating Tape Buffers

- From SGA (large pool) with `BACKUP_TAPE_IO_SLAVES` is `TRUE`.
- From PGA with `BACKUP_TAPE_IO_SLAVES` is `FALSE`.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Allocating Tape Buffers

If you make a backup to a tape device, the Oracle server allocates four buffers per channel for the tape writers (or readers if doing a restore). The Oracle server allocates these buffers only if the channel is a System Backup to Tape (SBT) channel. Typically, each tape buffer is 256 KB. To calculate the total size of buffers that are used during a backup or restore, multiply the buffer size by four, and then multiply this product by the number of channels.

As illustrated in the example shown in the slide, assume that you use one tape channel and each buffer is 256 KB. In this case, the total size of buffers that are used during a backup is as follows:

$$256 \text{ KB per buffer} \times 4 \text{ buffers per channel} \times 1 \text{ channel} = 1,024 \text{ KB}$$

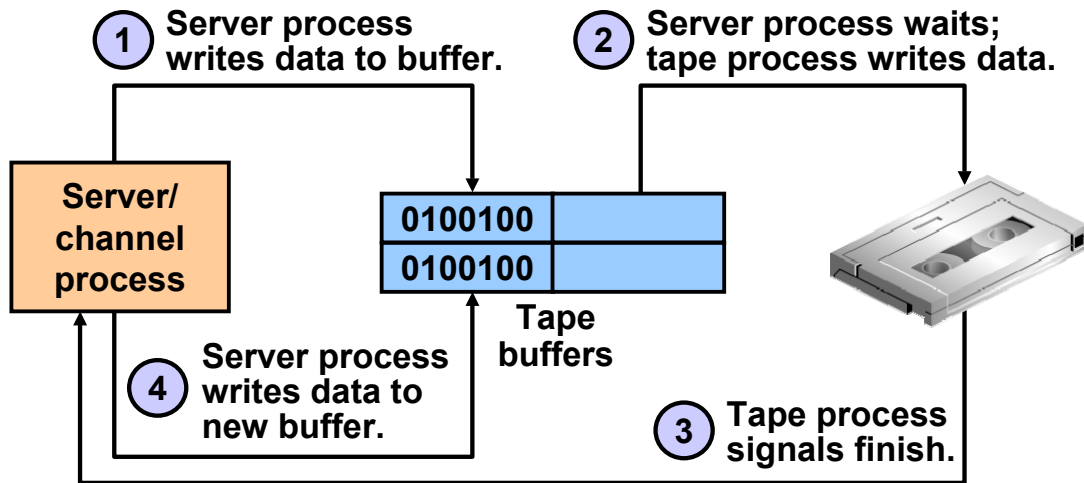
RMAN allocates the tape buffers in the System Global Area (SGA) or the Program Global Area (PGA), depending on whether I/O slaves are used. If the `BACKUP_TAPE_IO_SLAVES` initialization parameter is set to `TRUE`, RMAN allocates tape buffers from the shared pool or the large pool if the `LARGE_POOL_SIZE` initialization parameter is set. If you set the parameter to `FALSE`, RMAN allocates the buffers from the PGA. If you use I/O slaves, set the `LARGE_POOL_SIZE` initialization parameter to set aside SGA memory that is dedicated to holding these large memory allocations. By doing this, the RMAN I/O buffers do not compete with the library cache for shared pool memory.

Allocating Tape Buffers (continued)

Oracle recommends that you set the `BACKUP_TAPE_IO_SLAVES` initialization parameter to `TRUE`. In most circumstances, this will provide the best performance of backups to tape. Also, this setting is required in order to perform duplexed backups. Duplexed backups are covered in the lesson titled “Using RMAN to Create Backups.”

Comparing Synchronous and Asynchronous I/O

Synchronous I/O



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Comparing Synchronous and Asynchronous I/O

When RMAN reads or writes data, the I/O is either synchronous or asynchronous. When the I/O is synchronous, a server process can perform only one task at a time. When it is asynchronous, a server process can begin an I/O and then perform other tasks while waiting for the I/O to complete. It can also begin multiple I/O operations before waiting for the first to complete.

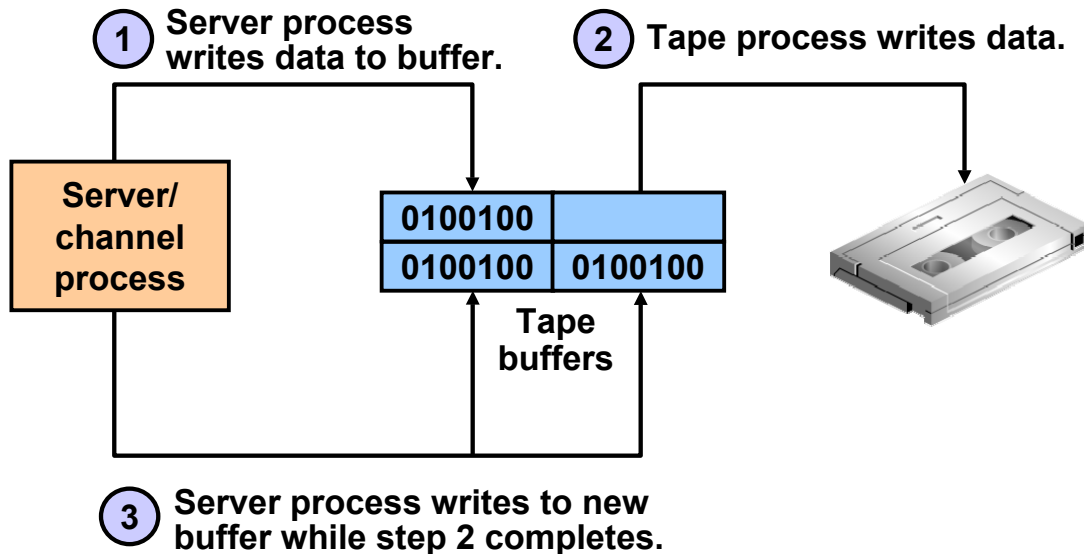
You can set initialization parameters that determine the type of I/O. If you set `BACKUP_TAPE_IO_SLAVES` to `TRUE`, the tape I/O is asynchronous. Otherwise, the I/O is synchronous.

The example in the slide shows synchronous I/O in a backup to tape. The following steps occur in a synchronous transfer:

1. A server process writes blocks to a tape buffer.
2. The tape process writes data to tape. The server process is idle while the media manager copies data from the Oracle buffers to the media manager's internal buffers.
3. The tape process relays to the server process that it has completed writing.
4. The server process can initiate a new task.

Comparing Synchronous and Asynchronous I/O

Asynchronous I/O



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Comparing Synchronous and Asynchronous I/O (continued)

Many operating systems support native asynchronous I/O and Oracle can take advantage of this feature whenever it is available. It is recommended that you always set `BACKUP_TAPE_IO_SLAVES` to `TRUE` when the platform supports it. On operating systems that do not support native asynchronous I/O, Oracle can simulate it by using special I/O slave processes that are dedicated to performing I/O on behalf of another process. You can control disk I/O slaves by setting the `DBWR_IO_SLAVES` parameter to a nonzero value. Oracle allocates four backup disk I/O slaves for any nonzero value of `DBWR_IO_SLAVES`.

The example in the slide illustrates asynchronous I/O in a backup to tape. The steps that occur in an asynchronous exchange are detailed below:

1. A server process writes blocks to a tape buffer.
2. The tape process writes data to the tape. While the tape process is writing, other server processes are free to process more input blocks and fill more output buffers.
3. The spawned server process writes to the tape buffers while the initial tape process writes to the tape.

Monitoring RMAN Job Performance

- The following views can be used to monitor backup and restore performance:
 - V\$BACKUP_SYNC_IO
 - V\$BACKUP_ASYNC_IO
- The following rows exist for a backup or restore:
 - One row for each data file
 - One aggregate data file row
 - One row for each backup piece
- Whether or not I/O is synchronous depends on how the controlling process views it.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring RMAN Job Performance

The maximum backup speed is limited by the available hardware. It is not possible to back up any faster than the aggregate tape bandwidth. One exception to this is if there are many empty blocks in the data files that need not be backed up.

One of the components of the backup system will be a bottleneck—which one depends on the relative speeds of the disk, tape drive, and any other transport components such as the network. As an example, if the bottleneck is the tape drive, and the tape is streaming, then the backup cannot possibly proceed any faster.

Note: If you have synchronous I/O and have set the BACKUP_DISK_IO_SLAVES initialization parameter to TRUE, I/O is displayed in V\$BACKUP_ASYNC_IO.

Asynchronous I/O Bottlenecks

- Use `V$BACKUP_ASYNC_IO` to monitor asynchronous I/O.
- The file that has the largest ratio of `LONG_WAITS` to `IO_COUNT` is probably the bottleneck.
 - `IO_COUNT`: Number of I/Os performed on the file
 - `LONG_WAITS`: Number of times the backup/restore process told the OS to wait until I/O was complete
- Wait times should be zero to avoid bottlenecks.
 - `SHORT_WAIT_TIME_TOTAL`
 - `LONG_WAIT_TIME_TOTAL`

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Asynchronous I/O Bottlenecks

You can use `V$BACKUP_ASYNC_IO` to monitor asynchronous I/O. The `LONG_WAITS` column shows the number of times the backup or restore process directed the operating system to wait until an I/O was complete. The `SHORT_WAITS` column shows the number of times the backup/restore process made an operating system call to poll for I/O completion in a nonblocking mode. On some platforms, the asynchronous I/O implementation may cause the calling process to wait for the I/O to complete while performing a nonblocking poll for I/O.

The simplest way to identify the bottleneck is to query `V$BACKUP_ASYNC_IO` for the data file that has the largest ratio for `LONG_WAITS` divided by `IO_COUNT`.

Synchronous I/O Bottlenecks

- Synchronous I/O is considered to be a bottleneck.
- Query the `DISCRETE_BYTES_PER_SECOND` column from `V$BACKUP_SYNC_IO` to view the I/O rate.
 - Compare this rate with the device's maximum rate.
 - If the rate is lower than what the device specifies, this is a tuning opportunity.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Synchronous I/O Bottlenecks

When using synchronous I/O, it can easily be determined how much time the backup jobs require because devices perform only one I/O task at a time. Oracle I/O uses a polling mechanism rather than an interrupt mechanism to determine when each I/O request completes. Because the backup or restore process is not immediately notified of I/O completion by the operating system, you cannot determine the duration of each I/O.

Use `V$BACKUP_SYNC_IO` to determine the source of backup or restore bottlenecks and to determine the progress of backup jobs. `V$BACKUP_SYNC_IO` contains rows when the I/O is synchronous to the process (or thread, on some platforms) that is performing the backup.

Channel Tuning

Use the `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands to:

- Limit the size of backup pieces
- Prevent RMAN from consuming too much disk bandwidth
- Determine the level of multiplexing for each channel
- Configure multiple disks, thus spreading the I/O activity across multiple devices.
- Configure multiple channels on the SBT device, allowing you to assign different data files to each one.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Channel Tuning

You can set various channel limit parameters that apply to operations that are performed by the allocated server session in the `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands.

The `MAXPIECESIZE` parameter specifies the maximum size of a backup piece. Use this parameter to make RMAN create multiple backup pieces in a backup set. RMAN creates each backup piece with a size that is no larger than the value that has been specified in the parameter.

The `RATE` parameter specifies the bytes per second that RMAN reads on the channel. This parameter is useful in preventing RMAN from consuming excessive disk bandwidth and degrading online transaction processing (OLTP) performance. For example, if each disk drive delivers 3 MB per second and you set `RATE=1500K`, some disk bandwidth will still be available to the online system.

The `MAXOPENFILES` parameter determines the maximum number of input files that a backup or copy can have open at a given time. If not set manually, the value defaults to 8. The level of RMAN multiplexing is partially determined by `MAXOPENFILES`. The level of multiplexing in turn determines how RMAN allocates disk buffers. Multiplexing is the number of input files that are simultaneously read from and then written into the same backup piece.

Channel Tuning (continued)

If you configure multiple channels for an SBT device, then you can specifically spread data files across those channels. Here is an example:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt;
  ALLOCATE CHANNEL c3 DEVICE TYPE sbt;
  BACKUP (DATAFILE 1,2,5 CHANNEL c1)
        (DATAFILE 4,6 CHANNEL c2)
        (DATAFILE 3,7,8 CHANNEL c3);
  BACKUP DATABASE NOT BACKED UP;
}
```

Tuning the BACKUP Command

- `MAXPIECESIZE` limits the size of each backup piece.
- `FILESERSET` prevents RMAN from reading from too many disks at once.
- `MAXOPENFILES` may inhibit streaming to tape if not set high enough.
- `BACKUP DURATION` decreases the amount of load on the system that the backup operation causes.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Tuning the BACKUP Command

The `MAXPIECESIZE` parameter specifies the maximum size of each backup piece created on the channel.

The `FILESERSET` parameter specifies the maximum number of files to place in a backup set. If you allocate only one channel, then you can use this parameter to make RMAN create multiple backup sets. For example, if you have 50 input data files and two channels, you can set `FILESERSET=5` to create 10 backup sets. This strategy can prevent you from splitting a backup set among multiple tapes.

The `MAXOPENFILES` parameter setting depends on your disk subsystem characteristics. If you use ASM, then set it to 1 or 2. Otherwise, if your data is not striped, then you may want to set this higher. To gain performance, increase either the number of files per backup set, or this parameter. If you are not using ASM or striping of any kind, then try increasing `MAXOPENFILES`.

Tuning the BACKUP Command (continued)

The BACKUP DURATION option of the BACKUP command can be used in different ways. If you specify a shorter duration than needed for the backup to complete, then you can use this to keep the backup activity inside a specific time window. In specific cases, the partial backup that does complete is not lost.

Also, this option has two modifiers:

- **MINIMIZE TIME:** The backup runs as fast as possible.
- **MINIMIZE LOAD:** The backup attempts to use the full amount of time available in the window. This reduces load on the system.

Tuning RMAN Backup Performance

To tune RMAN backup performance, follow these steps :

1. Remove `RATE` settings from configured and allocated channels.
2. Set `DBWR_IO_SLAVES` if you use synchronous disk I/O.
3. Set `LARGE_POOL_SIZE`.
4. Tune RMAN tape streaming performance bottlenecks.
5. Query `V$` views to identify bottlenecks.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Tuning RMAN Backup Performance

Follow this set of steps to obtain the best backup performance:

1. Remove `RATE` settings from configured and allocated channels. The `RATE` parameter is used to set the maximum number of bytes (default), kilobytes (K), megabytes (M), or gigabytes (G) that RMAN reads each second on the channel. It sets an upper limit for bytes read so that RMAN does not consume too much disk bandwidth and degrade performance. If your backup is not streaming to tape, ensure that the `RATE` parameter is not set on the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command.
2. Set `DBWR_IO_SLAVES` if you use synchronous disk I/O. If your disk does not support asynchronous I/O, then try setting the `DBWR_IO_SLAVES` initialization parameter to a nonzero value. Any nonzero value for `DBWR_IO_SLAVES` causes a fixed number (four) of disk I/O slaves to be used for backup and restore, simulating asynchronous I/O. If I/O slaves are used, I/O buffers are obtained from the SGA. The large pool is used if configured. Otherwise, the shared pool is used.
Note: By setting `DBWR_IO_SLAVES`, the database writer processes will use slaves as well. You may need to increase the value of the `PROCESSES` initialization parameter.
3. Set `LARGE_POOL_SIZE` as described on the next page.
4. Tune RMAN tape streaming performance bottlenecks as described later in the lesson.
5. Use `V$` views as described earlier in the lesson.

Setting LARGE_POOL_SIZE

- If LARGE_POOL_SIZE is not set, the Oracle server tries to get memory from the shared pool.
- If LARGE_POOL_SIZE is not big enough, the server does not allocate buffers from the shared pool.
- If the server cannot get enough memory, it allocates buffers from the local process memory.
- The Oracle server writes a message to the alert log indicating that synchronous I/O is used for this backup.

```
ksfqxcrc: failure to allocate shared memory means sync
I/O will be used whenever async I/O to file not
supported natively
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Setting LARGE_POOL_SIZE

The requests for contiguous memory allocations from the shared pool are small, usually under 5 KB in size. It is possible that a request for a large contiguous memory allocation can fail or require significant memory housekeeping to release the required amount of contiguous memory. The large pool may be able to satisfy the memory request. The large pool does not have a least recently used list, so Oracle does not attempt to age memory out of the large pool.

Use the LARGE_POOL_SIZE initialization parameter to configure the large pool. Query V\$SGASTAT.POOL to see in which pool (shared pool or large pool) the memory for an object resides. The suggested value for LARGE_POOL_SIZE is calculated as:

$$\#_of_allocated_channels * (16 \text{ MB} + (4 * size_of_tape_buffer))$$

For backups to disk, the tape buffer is obviously 0, so set LARGE_POOL_SIZE to 16 MB. For tape backups, the size of a single tape buffer is defined by the RMAN channel parameter BLKSIZE, which defaults to 256 KB. Assume a case in which you are backing up to two tape drives. If the tape buffer size is 256 KB, then set LARGE_POOL_SIZE to 18 MB. If you increase BLKSIZE to 512 KB, then increase LARGE_POOL_SIZE to 20 MB.

Note: The large pool is used only for disk buffers when DBWR_IO_SLAVES > 0 and for tape buffers when BACKUP_TAPE_IO_SLAVES = TRUE. If you are using Automatic Shared Memory Management, the large pool is sized automatically in response to system workload.

Tuning RMAN Tape Streaming Performance Bottlenecks

- Use `BACKUP . . . VALIDATE` to determine whether tape streaming or disk I/O is the bottleneck.
- Use multiplexing to improve tape streaming with disk bottlenecks.
- Use incremental backups to improve backup performance with tape bottlenecks.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Tuning RMAN Tape Streaming Performance Bottlenecks

To identify and remedy bottlenecks that affect RMAN's performance on tape backups, perform the following actions :

- Use `BACKUP . . . VALIDATE` to determine whether tape streaming or disk I/O is the bottleneck in a given backup job. Compare the time required to run backup tasks with the time required to run `BACKUP VALIDATE` of the same tasks. `BACKUP VALIDATE` of a backup to tape performs the same disk reads as a real backup but performs no tape I/O. If the time required for the `BACKUP VALIDATE` to tape is significantly less than the time required for a real backup to tape, then writing to tape is the likely bottleneck.
- Use multiplexing to improve tape streaming with disk bottlenecks. In some situations when RMAN is performing a backup to tape, it may not be able to send data blocks to the tape drive fast enough to support streaming. For example, during an incremental backup, RMAN backs up only blocks changed since a previous data file backup as part of the same strategy. If you do not enable change tracking, RMAN must scan entire data files for changed blocks, and fill output buffers as it finds such blocks. If there are not many changed blocks, RMAN may not fill output buffers fast enough to keep the tape drive streaming. You can improve performance by increasing the degree of multiplexing used for backups. This increases the rate at which RMAN fills tape buffers, which makes it more likely that buffers are sent to the media manager fast enough to maintain streaming.

Tuning RMAN Tape Streaming Performance Bottlenecks (continued)

- Use incremental backups to improve backup performance with tape bottlenecks. If writing to tape is the source of a bottleneck for your backups, consider using incremental backups as part of your backup strategy. Incremental level 1 backups write only the changed blocks from data files to tape, so that any bottleneck on writing to tape has less impact on your overall backup strategy. In particular, if tape drives are not locally attached to the node running the database being backed up, then incremental backups can be faster.

Quiz

Select which statements are true about RMAN tuning:

1. You can configure parallel backups by setting the `PARALLELISM` option of the `CONFIGURE` command to greater than 1 or by manually allocating multiple channels.
2. You can stripe a single backup set across multiple channels to improve performance.
3. Whenever you improve the speed of the backup operation, you also automatically improve the speed of the restore and recover operations.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Quiz

You can never have RMAN bottlenecks because the Tuning Advisor fixes them automatically.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Summary

In this lesson, you should have learned how to:

- Monitor the progress of RMAN jobs
- Configure RMAN appropriately for asynchronous I/O
- Configure RMAN multiplexing so as to keep tape drives streaming efficiently
- Evaluate the balance between speed of backup versus speed of recovery
- Explain the effect of the following parameters on RMAN performance: MAXPIECESIZE, FILESPERSET, MAXOPENFILES
- Explain how the RMAN BACKUP DURATION option can cause backups to either execute faster or take longer (freeing up resources for other processing)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 8 Overview: Monitoring and Tuning RMAN

This practice covers the following topics:

- Monitoring RMAN jobs
- Using EM to monitor RMAN

ORACLE

Copyright © 2009, Oracle. All rights reserved.

9

Diagnosing the Database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Detect and repair database corruption
- Handle block corruption
- Set up Automatic Diagnostic Repository
- Run health checks

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Data Recovery Advisor

> **Data Recovery Ad.**
Block Corruption
ADR
Health Monitor

- Fast detection, analysis, and repair of failures
- Minimizing disruptions for users
- Down-time and run-time failures
- User interfaces:
 - EM GUI interface (several paths)
 - RMAN command line
- Supported database configurations:
 - Single-instance
 - Not RAC
 - Supporting failover to standby, but not analysis and repair of standby databases



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Data Recovery Advisor

The Data Recovery Advisor automatically gathers data failure information when an error is encountered. In addition, it can proactively check for failures. In this mode, it can potentially detect and analyze data failures before a database process discovers the corruption and signals an error. (Note that repairs are always under human control.)

Data failures can be very serious. For example, if your current log files are missing, you cannot start your database. Some data failures (such as block corruptions in data files) are not catastrophic, in that they do not take the database down or prevent you from starting the Oracle instance. The Data Recovery Advisor handles both cases: the one when you cannot start up the database (because some required database files are missing, inconsistent, or corrupted) and the one when file corruptions are discovered during run time.

User Interfaces

The Data Recovery Advisor is available from Enterprise Manager (EM) Database Control and Grid Control. When failures exist, there are several ways to access the Data Recovery Advisor. The following examples all begin on the Database Instance home page:

- Availability tabbed page > Perform Recovery > Advise and Recover
- Active Incidents link > on the Support Workbench “Problems” page: Checker Findings tabbed page > Launch Recovery Advisor
- Database Instance Health > click the specific link, for example, ORA 1578 in the Incidents section > Support Workbench, Problems Detail page > Data Recovery Advisor
- Database Instance Health > Related Links section: Support Workbench > Checker Findings tabbed page: Launch Recovery Advisor
- Related Link: Advisor Central > Advisors tabbed page: Data Recovery Advisor
- Related Link: Advisor Central > Checkers tabbed page: Details > Run Detail tabbed page: Launch Recovery Advisor

You can also use it via the RMAN command-line. For example:

```
rman target / nocatalog
rman> list failure all;
```

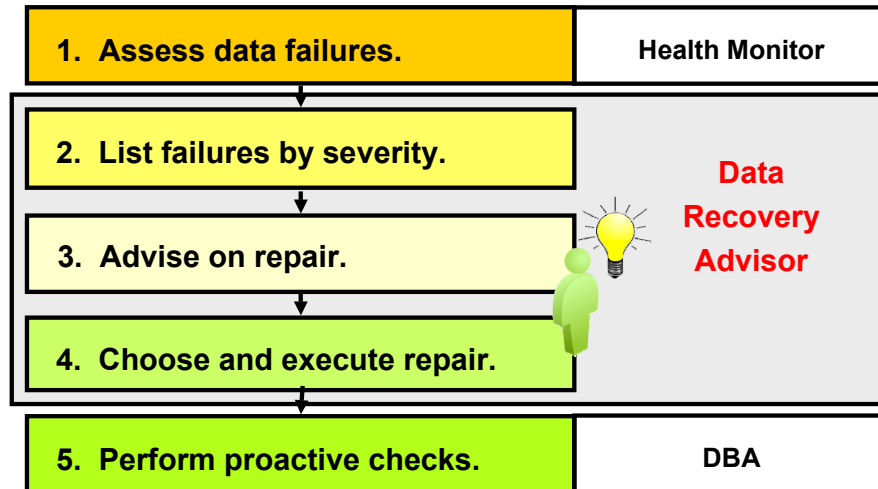
Supported Database Configurations

In the current release, the Data Recovery Advisor supports single-instance databases. Oracle Real Application Clusters (RAC) databases are not supported.

The Data Recovery Advisor cannot use blocks or files transferred from a standby database to repair failures on a primary database. Also, you cannot use the Data Recovery Advisor to diagnose and repair failures on a standby database. However, the Data Recovery Advisor does support failover to a standby database as a repair option (as mentioned above).

Data Recovery Advisor

Reducing down time by eliminating confusion:



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Data Recovery Advisor

The automatic diagnostic workflow in Oracle Database 11g performs as follows. With the Data Recovery Advisor, you only need to initiate an advice and a repair.

1. The Health Monitor automatically executes checks and logs failures and their symptoms as “findings” into the Automatic Diagnostic Repository (ADR).
2. The Data Recovery Advisor consolidates findings into failures. It lists the results of previously executed assessments with failure severity (critical or high).
3. When you ask for repair advice on a failure, the Data Recovery Advisor maps failures to automatic and manual repair options, checks basic feasibility, and presents you with the repair advice.
4. You can choose to manually execute a repair or request the Data Recovery Advisor to do it for you.
5. In addition to the automatic, primarily “reactive” checks of the Health Monitor and Data Recovery Advisor, Oracle recommends to additionally use the `VALIDATE` command as a “proactive” check.

Data Failures

The screenshot displays the Oracle Enterprise Manager 11g Database Control interface. At the top, the title bar reads "ORACLE Enterprise Manager 11g Database Control". Below this, the "Database Instance: orcl" is selected. The "Information" tab is active, showing a list of items: "1. Database Failures - 1" and "2. Current Status - MOUNTED". The "Perform Recovery" section is expanded, showing "Oracle Advised Recovery" and "User Directed Recovery". In the "Oracle Advised Recovery" section, a red box highlights the "Advise and Recover" button. Below this, the "Failures Detected" section shows "Critical: 0 High: 1 Low: 0" and a "Failure Description" stating "One or more non-system datafiles are missing". The "User Directed Recovery" section includes a "Recovery Scope" dropdown set to "Whole Database" and a "Recover" button. The "Operation Type" section lists three options: "Recover to the current time or a previous point-in-time", "Restore all datafiles", and "Recover from previously restored datafiles". The "Overview" section on the right lists several recovery actions: "Recover database failures as advised by Oracle", "Restore and/or recover the entire database or selected objects", "Restore files to a new location", "Recover tablespaces to a point-in-time based on a timestamp, system change number (SCN), or log sequence number", "Recover datafile data blocks that are marked as corrupted, or based on datafile block IDs or tablespace block addresses", and "Flashback database, tables, or transactions to a specific system change number (SCN) or timestamp". The bottom of the interface features a red banner with the "ORACLE" logo and the text "Copyright © 2009, Oracle. All rights reserved."

Data Failures

Data failures are detected by checks, which are diagnostic procedures that assess the health of the database or its components. Each check can diagnose one or more failures, which are mapped to a repair.

Checks can be reactive or proactive. When an error occurs in the database, “reactive checks” are automatically executed. You can also initiate “proactive checks”, for example, by executing the `VALIDATE DATABASE` command.

In Enterprise Manager, select Availability > Perform Recovery, or click the Perform Recovery button, if you find your database in a “down” or “mounted” state.

Data Failure: Examples

- Not accessible components, for example:
 - Missing data files at the OS level
 - Incorrect access permissions
 - Offline tablespace, and so on
- Physical corruptions, such as block checksum failures or invalid block header field values
- Logical corruptions, such as inconsistent dictionary, corrupt row piece, corrupt index entry, or corrupt transaction
- Inconsistencies, such as control file is older or newer than the data files and online redo logs
- I/O failures, such as a limit on the number of open files exceeded, channels inaccessible, network or I/O error



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Data Failure: Examples

The Data Recovery Advisor can analyze failures and suggest repair options for issues, as outlined in the slide.

Data Recovery Advisor RMAN Command-Line Interface

RMAN Command	Action
LIST FAILURE	Lists previously executed failure assessment
ADVISE FAILURE	Displays recommended repair option
REPAIR FAILURE	Repairs and closes failures (after ADVISE in the same RMAN session)
CHANGE FAILURE	Changes or closes one or more failures

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Data Recovery Advisor: RMAN Command-Line Interface

If you suspect or know that a database failure has occurred, then use the `LIST FAILURE` command to obtain information about these failures. You can list all or a subset of failures and restrict output in various ways. Failures are uniquely identified by failure numbers. Note that these numbers are not consecutive, so gaps between failure numbers have no significance.

The `ADVISE FAILURE` command displays a recommended repair option for the specified failures. It prints a summary of the input failure and implicitly closes all open failures that are already fixed. The default behavior when no option is used is to advise on all the `CRITICAL` and `HIGH` priority failures that are recorded in ADR.

The `REPAIR FAILURE` command is used after an `ADVISE FAILURE` command **within the same RMAN session**. By default, the command uses the single, recommended repair option of the last `ADVISE FAILURE` execution in the current session. If none exists, the `REPAIR FAILURE` command initiates an implicit `ADVISE FAILURE` command. After completing the repair, the command closes the failure.

The `CHANGE FAILURE` command changes the failure priority or closes one or more failures. You can change a failure priority only for `HIGH` or `LOW` priorities. Open failures are closed implicitly when a failure is repaired. However, you can also explicitly close a failure.

Listing Data Failures

The RMAN `LIST FAILURE` command lists previously executed failure assessment.

- Including newly diagnosed failures
- Removing closed failures (by default)



Syntax:

```
LIST FAILURE
[ ALL | CRITICAL | HIGH | LOW | CLOSED |
  failnum[,failnum,...] ]
[ EXCLUDE FAILURE failnum[,failnum,...] ]
[ DETAIL ]
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Listing Data Failures

The RMAN `LIST FAILURE` command lists failures. If the target instance uses a recovery catalog, it can be in `STARTED` mode, otherwise it must be in `MOUNTED` mode. The `LIST FAILURE` command does not initiate checks to diagnose new failures; rather, it lists the results of previously executed assessments. Repeatedly executing the `LIST FAILURE` command revalidates all existing failures. If the database diagnoses new ones (between command executions), they are displayed. If a user manually fixes failures, or if transient failures disappear, then the Data Recovery Advisor removes these failures from the `LIST FAILURE` output. The following is a description of the syntax:

- `failnum`: Number of the failure to display repair options for
- `ALL`: List failures of all priorities.
- `CRITICAL`: List failures of `CRITICAL` priority and `OPEN` status. These failures require immediate attention, because they make the whole database unavailable (for example, a missing control file).
- `HIGH`: List failures of `HIGH` priority and `OPEN` status. These failures make a database partly unavailable or unrecoverable; so they should be repaired quickly (for example, missing archived redo logs).
- `LOW`: List failures of `LOW` priority and `OPEN` status. Failures of a low priority can wait until more important failures are fixed.

Listing Data Failures (continued)

- **CLOSED:** List only closed failures.
- **EXCLUDE FAILURE:** Exclude the specified list of failure numbers from the list.
- **DETAIL:** List failures by expanding the consolidated failure. For example, if there are multiple block corruptions in a file, the **DETAIL** option lists each one of them.

See the *Oracle Database Backup and Recovery Reference* for details of the command syntax.

Advising on Repair

The RMAN ADVISE FAILURE command:

- Displays a summary of input failure list
- Includes a warning, if new failures appeared in ADR
- Displays a manual checklist
- Lists a single recommended repair option
- Generates a repair script (for automatic or manual repair)

```
. . .
Repair script:
  /u01/app/oracle/diag/rdbms/orcl/orcl/hm/reco_2979
  128860.hm
RMAN>
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Advising on Repair

The RMAN ADVISE FAILURE command displays a recommended repair option for the specified failures. The ADVISE FAILURE command prints a summary of the input failure. The command implicitly closes all open failures that are already fixed.

The default behavior (when no option is used) is to advise on all the CRITICAL and HIGH priority failures that are recorded in Automatic Diagnostic Repository (ADR). If a new failure has been recorded in ADR since the last LIST FAILURE command, this command includes a WARNING before advising on all CRITICAL and HIGH failures.

Two general repair options are implemented: no-data-loss and data-loss repairs.

When the Data Recovery Advisor generates an automated repair option, it generates a script that shows you how RMAN plans to repair the failure. If you do not want the Data Recovery Advisor to automatically repair the failure, then you can use this script as a starting point for your manual repair. The operating system (OS) location of the script is printed at the end of the command output. You can examine this script, customize it (if needed), and also execute it manually if, for example, your audit trail requirements recommend such an action.

Syntax

```
ADVISE FAILURE
[ ALL | CRITICAL | HIGH | LOW | failnum[,failnum,...] ]
[ EXCLUDE FAILURE failnum [,failnum,...] ]
```

Executing Repairs

The RMAN REPAIR FAILURE command:

- Follows the ADVISE FAILURE command
- Repairs the specified failure
- Closes the repaired failure

Syntax:

```
REPAIR FAILURE
[USING ADVISE OPTION integer]
[ { {NOPROMPT | PREVIEW}}...]
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Executing Repairs

This command should be used after an ADVISE FAILURE command in the same RMAN session. By default (with no option), the command uses the single, recommended repair option of the last ADVISE FAILURE execution in the current session. If none exists, the REPAIR FAILURE command initiates an implicit ADVISE FAILURE command.

With USING ADVISE OPTION *integer*, you specify your desired repair option by its option number (from the ADVISE FAILURE command); this is not the failure number.

By default, you are asked to confirm the command execution because you may be requesting substantial changes that take time to complete. During the execution of a repair, the output of the command indicates what phase of the repair is being executed.

After completing the repair, the command closes the failure.

You cannot run multiple concurrent repair sessions. However, concurrent REPAIR ... PREVIEW sessions are allowed.

- PREVIEW means: Do not execute the repairs; instead, display the previously generated RMAN script with all repair actions and comments.
- NOPROMPT means: Do not ask for confirmation.

Classifying (and Closing) Failures

The RMAN `CHANGE FAILURE` command:

- Changes the failure priority (except for `CRITICAL`)
- Closes one or more failures

Example:

```
RMAN> change failure 5 priority low;
List of Database Failures
=====
Failure ID Priority Status      Time Detected Summary
-----
5          HIGH    OPEN      20-DEC-06    one or more
            datafiles are missing
Do you really want to change the above failures (enter YES or
NO)? yes
changed 1 failures to LOW priority
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Classifying (and Closing) Failures

The `CHANGE FAILURE` command is used to change the failure priority or close one or more failures.

Syntax

```
CHANGE FAILURE
{ ALL | CRITICAL | HIGH | LOW | failnum[,failnum,...] }
[ EXCLUDE FAILURE failnum[,failnum,...] ]
{ PRIORITY {CRITICAL | HIGH | LOW} |
CLOSE } – change status of the failure(s) to closed
[ NOPROMPT ] – do not ask user for a confirmation
```

A failure priority can be changed only from `HIGH` to `LOW` and from `LOW` to `HIGH`. It is an error to change the priority level of `CRITICAL`. (One reason why you may want to change a failure from `HIGH` to `LOW` is to avoid seeing it on the default output list of the `LIST FAILURE` command. For example, if a block corruption has `HIGH` priority, you may want to temporarily change it to `LOW` if the block is in a little-used tablespace.)

Open failures are closed implicitly when a failure is repaired. However, you can also explicitly close a failure. This involves a reevaluation of all other open failures, because some of them may become irrelevant as the result of the closure of the failure.

By default, the command asks the user to confirm a requested change.

Data Recovery Advisor Views

Querying V\$ views:

- V\$IR_FAILURE: List of all failures, including closed ones (result of the LIST FAILURE command)
- V\$IR_MANUAL_CHECKLIST: List of manual advice (result of the ADVISE FAILURE command)
- V\$IR_REPAIR: List of repairs (result of the ADVISE FAILURE command)
- V\$IR_FAILURE_SET: Cross-reference of failure and advice identifiers



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Data Recovery Advisor Views

See the Oracle Database Reference for details of the dynamic data dictionary views that the Data Recovery Advisor uses.

Best Practice: Proactive Checks

Invoking proactive health check of the database and its components:

- Health Monitor or RMAN `VALIDATE DATABASE` command
- Checking for logical and physical corruption
- Findings logged in the ADR



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Best Practice: Proactive Checks

For very important databases, you may want to execute additional proactive checks (possibly daily during low peak interval periods). You can schedule periodic health checks through the Health Monitor or by using the RMAN `VALIDATE` command. In general, when a reactive check detects failure(s) in a database component, you may want to execute a more complete check of the affected component.

The RMAN `VALIDATE DATABASE` command is used to invoke health checks for the database and its components. It extends the existing `VALIDATE BACKUPSET` command. Any problem detected during validation is displayed to you. Problems initiate the execution of a failure assessment. If a failure is detected, it is logged into ADR as a finding. You can use the `LIST FAILURE` command to view all failures recorded in the repository.

The `VALIDATE` command supports validation of individual backup sets and data blocks. In a physical corruption, the database does not recognize the block at all. In a logical corruption, the contents of the block are logically inconsistent. By default, the `VALIDATE` command checks for physical corruption only. You can specify `CHECK LOGICAL` to check for logical corruption as well.

Block corruptions can be divided into interblock corruption and intrablock corruption. In intrablock corruption, the corruption occurs within the block itself and can be either physical or logical corruption. In interblock corruption, the corruption occurs between blocks and can be only logical corruption. The `VALIDATE` command checks for intrablock corruptions only.

What Is Block Corruption?

Data Recovery Ad.
> Block Corruption
ADR
Health Monitor

- Whenever a block is read or written, a consistency check is performed.
 - Block version
 - DBA (data block address) value in cache as compared to the DBA value in the block buffer
 - Block-checksum, if enabled
- A corrupt block is identified as being one of the following:
 - Media corrupt
 - Logically (or software) corrupt

ORACLE

Copyright © 2009, Oracle. All rights reserved.

What Is Block Corruption?

A corrupted data block is a block that is not in a recognized Oracle format, or whose contents are not internally consistent. Typically, corruptions are caused by faulty hardware or operating system problems. The Oracle database identifies corrupt blocks as either “logically corrupt” or “media corrupt.” If it is logically corrupt, there is an Oracle internal error. Logically corrupt blocks are marked corrupt by the Oracle database after it detects the inconsistency. If it is media corrupt, the block format is not correct; the information in the block does not make any sense after being read from disk.

As you just learned, a number of data failures and corruptions can be repaired with the Data Recovery Advisor. Now you learn about a manual way to diagnose and repair corruptions.

You can repair a media corrupt block by recovering the block or dropping the database object that contains the corrupt block, or both. If media corruption is due to faulty hardware, the problem will not be completely resolved until the hardware fault is corrected.

Block Corruption Symptoms: ORA-01578

The error ORA-01578: "ORACLE data block corrupted (file # %s, block # %s)":

- Is generated when a corrupted data block is found
- Always returns the relative file number and block number
- Is returned to the session that issued the query being performed when the corruption was discovered
- Appears in the `alert.log` file

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Block Corruption Symptoms: ORA-01578

Usually, the ORA-01578 error is the result of a hardware problem. If the ORA-01578 error is always returned with the same arguments, it is most likely a media corrupt block.

If the arguments change each time, there may be a hardware problem, and you should have the memory and page space checked, and the I/O subsystem checked for bad controllers.

Note: ORA-01578 returns the relative file number, but the accompanying ORA-01110 error displays the absolute file number.

How to Handle Corruption

- Check the alert log and operating system log file.
- Use available diagnostic tools to find out the type of corruption.
- Determine whether the error persists by running checks multiple times.
- Recover data from the corrupted object if necessary.
- Resolve any hardware issues:
 - Memory boards
 - Disk controllers
 - Disks
- Recover or restore data from the corrupt object if necessary.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

How to Handle Corruption

Always try to find out whether the error is permanent. Run the `ANALYZE` command multiple times or, if possible, perform a shutdown and a startup and try again to perform the operation that failed earlier. Find out whether there are more corruptions. If you encounter one, there may be other corrupted blocks, as well.

Hardware failures should be addressed immediately. When you encounter hardware problems, the vendor should be contacted and the machine should be checked and fixed before continuing. A full hardware diagnostics session should be run.

Many types of hardware failures are possible:

- Faulty I/O hardware or firmware
- Operating system I/O or caching problem
- Memory or paging problems
- Disk repair utilities

Setting Parameters to Detect Corruption

Name	Help	Revisions	Value	Comments	Type	Basic	Modified	Dynamic	Category
db_block_checking			FALSE	Prevent memory and data corruption				✓	Diagnostics and Statistics
db_block_checksum			FALSE		String			✓	Diagnostics and Statistics
db_block_size			LOW		Integer	✓	✓		Memory
db_block_size			MEDIUM		Integer	✓	✓		Memory
db_block_size			TRUE		String			✓	Memory
db_block_size			FULL		String			✓	Memory
...									
db_block_checksum			TYPICAL	Detect I/O storage, disk corruption				✓	Diagnostics and Statistics
db_block_size			OFF		Integer	✓	✓		Memory
db_block_size			FALSE		Integer	✓	✓		Memory
db_block_size			TYPICAL		String			✓	Memory
db_block_size			TRUE		String			✓	Memory
db_block_size			FULL		Big			✓	Memory
...									
db_lost_write_protect			TYPICAL	Detect nonpersistent writes on physical standby					
db_name			NONE		String	✓	✓		Database Identification
db_name			TYPICAL		String				Database Identification
db_name			FULL		String				Database Identification
...									
db_ultra_safe			OFF	Specify defaults for corruption detection					Miscellaneous

EM > Server > Initialization Parameters

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Setting Parameters to Detect Corruption

You can use the DB_ULTRA_SAFE parameter for easy manageability. It affects the default values of the following parameters:

- DB_BLOCK_CHECKING, which initiates checking of database blocks. This check can often prevent memory and data corruption. (Default: FALSE, recommended: FULL)
- DB_BLOCK_CHECKSUM, which initiates the calculation and storage of a checksum in the cache header of every data block when writing it to disk. Checksums assist in detecting corruption caused by underlying disks, storage systems, or I/O systems. (Default: TYPICAL, recommended: TYPICAL)
- DB_LOST_WRITE_PROTECT, which initiates checking for “lost writes.” Data block lost writes occur on a physical standby database, when the I/O subsystem signals the completion of a block write, which has not yet been completely written in persistent storage. Of course, the write operation has been completed on the primary database. (Default: TYPICAL, recommended: TYPICAL)

If you set any of these parameters explicitly, your values remain in effect. The DB_ULTRA_SAFE parameter (which is new in Oracle Database 11g) changes only the **default** values for these parameters.

Setting Parameters to Detect Corruption

DB_ULTRA_SAFE	OFF	DATA_ONLY	DATA_AND_INDEX
DB_BLOCK_CHECKING	OFF or FALSE	MEDIUM	FULL or TRUE
DB_BLOCK_CHECKSUM	TYPICAL	FULL	FULL
DB_LOST_WRITE_PROTECT	TYPICAL	TYPICAL	TYPICAL

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Setting Parameters to Detect Corruption (continued)

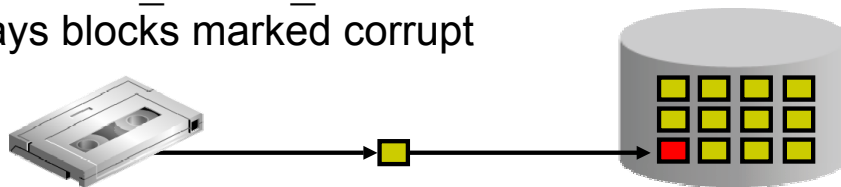
Depending on your system's tolerance for block corruption, you can intensify the checking for block corruption. Enabling the DB_ULTRA_SAFE parameter (default: OFF) results in increased system overhead because of these more intensive checks. The amount of overhead is related to the number of blocks changed per second, so it cannot be easily quantified. For a "high-update" application, you can expect a significant increase in CPU, likely in the ten-to-twenty percent range, but possibly higher. This overhead can be alleviated by allocating additional CPUs.

- When the DB_ULTRA_SAFE parameter is set to DATA_ONLY, the DB_BLOCK_CHECKING parameter is set to MEDIUM. This checks that data in a block are logically self-consistent. Basic block header checks are performed after block contents change in memory (for example, after UPDATE or INSERT commands, on-disk reads, or interinstance block transfers in Oracle RAC). This level of checks includes semantic block checking for all non-index-organized table blocks.
- When the DB_ULTRA_SAFE parameter is set to DATA_AND_INDEX, the DB_BLOCK_CHECKING parameter is set to FULL. In addition to the preceding checks, semantic checks are executed for index blocks (that is, blocks of subordinate objects that can actually be dropped and reconstructed when faced with corruption).
- When the DB_ULTRA_SAFE parameter is set to DATA_ONLY or DATA_AND_INDEX, the DB_BLOCK_CHECKSUM parameter is set to FULL and the DB_LOST_WRITE_PROTECT parameter is set to TYPICAL.

Block Media Recovery

Block media recovery:

- Lowers the mean time to recover (MTTR)
- Increases availability during media recovery
 - The data file remains online during recovery
 - Only blocks being recovered are inaccessible
- Is invoked using the **RMAN RECOVER . . . BLOCK** command
 - Restores blocks using flashback logs and full or level 0 backups
 - Media recovery is performed using redo logs
- The **V\$DATABASE_BLOCK_CORRUPTION** view displays blocks marked corrupt



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Block Media Recovery

In most cases, the database marks a block as media corrupt and then writes it to disk when the corruption is first encountered. No subsequent read of the block will be successful until the block is recovered. You can only perform block recovery on blocks that are marked corrupt or fail a corruption check. Block media recovery is performed using the **RMAN RECOVER . . . BLOCK** command. By default, RMAN searches the flashback logs for good copies of the blocks, and then searches for the blocks in full or level 0 incremental backups. When RMAN finds good copies, it restores them and performs media recovery on the blocks. Block media recovery can only use redo logs for media recovery, not incremental backups.

The **V\$DATABASE_BLOCK_CORRUPTION** view displays blocks marked corrupt by database components such as RMAN commands, **ANALYZE**, **dbv**, SQL queries, and so on. The following types of corruption result in rows added to this view:

- **Physical/Media corruption:** The database does not recognize the block: the checksum is invalid, the block contains all zeros, or the block header is fractured. Physical corruption checking is enabled by default.
- **Logical corruption:** The block has a valid checksum, the header and footer match, but the contents are inconsistent. Block media recovery cannot repair logical block corruption. Logical corruption checking is disabled by default. You can turn it on by specifying the **CHECK LOGICAL** option of the **BACKUP**, **RESTORE**, **RECOVER**, and **VALIDATE** commands.

Prerequisites for Block Media Recovery

- The target database must be in ARCHIVELOG mode.
- The backups of the data files containing the corrupt blocks must be full or level 0 backups.
 - Proxy copies must be restored to a non-default location before they can be used.
- RMAN can use only archived redo logs for the recovery.
- The corrupted data block can be restored from Flashback Logs if available.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Prerequisites for Block Media Recovery

The following prerequisites apply to the `RECOVER . . . BLOCK` command:

- The target database must run in ARCHIVELOG mode and be open or mounted with a current control file.
- The backups of the data files containing the corrupt blocks must be full or level 0 backups and not proxy copies. If only proxy copy backups exist, then you can restore them to a non-default location on disk, in which case RMAN considers them data file copies and searches them for blocks during block media recovery.
- RMAN can use only archived redo logs for the recovery. RMAN cannot use level 1 incremental backups. Block media recovery cannot survive a missing or inaccessible archived redo log, although it can sometimes survive missing redo records.
- Flashback Database must be enabled on the target database for RMAN to search the flashback logs for good copies of corrupt blocks. If flashback logging is enabled and contains older, uncorrupted versions of the corrupt blocks, then RMAN can use these blocks, possibly speeding up the recovery.

The RECOVER...BLOCK Command

The RMAN RECOVER...BLOCK command:

- Identifies the backups containing the blocks to recover
- Reads the backups and accumulates requested blocks into in-memory buffers
- Manages the block media recovery session by reading the archive logs from backup if necessary

```
RECOVER DATAFILE 6 BLOCK 3; Recover a single block

RECOVER                                Recover multiple blocks
DATAFILE 2 BLOCK 43                    in multiple data files
DATAFILE 2 BLOCK 79
DATAFILE 6 BLOCK 183;

RECOVER CORRUPTION LIST; Recover all blocks logged in
                        V$DATABASE_BLOCK_CORRUPTION
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recovering Individual Blocks

Before block recovery can take place, you must identify the corrupt blocks. Typically, block corruption is reported in the following locations:

- Results of the LIST FAILURE, VALIDATE, or BACKUP ... VALIDATE command
- The V\$DATABASE_BLOCK_CORRUPTION view
- Error messages in standard output
- The alert log and user trace files (identified in the V\$DIAG_INFO view)
- Results of the SQL ANALYZE TABLE and ANALYZE INDEX commands
- Results of the DBVERIFY utility

For example, you may discover the following messages in a user trace file:

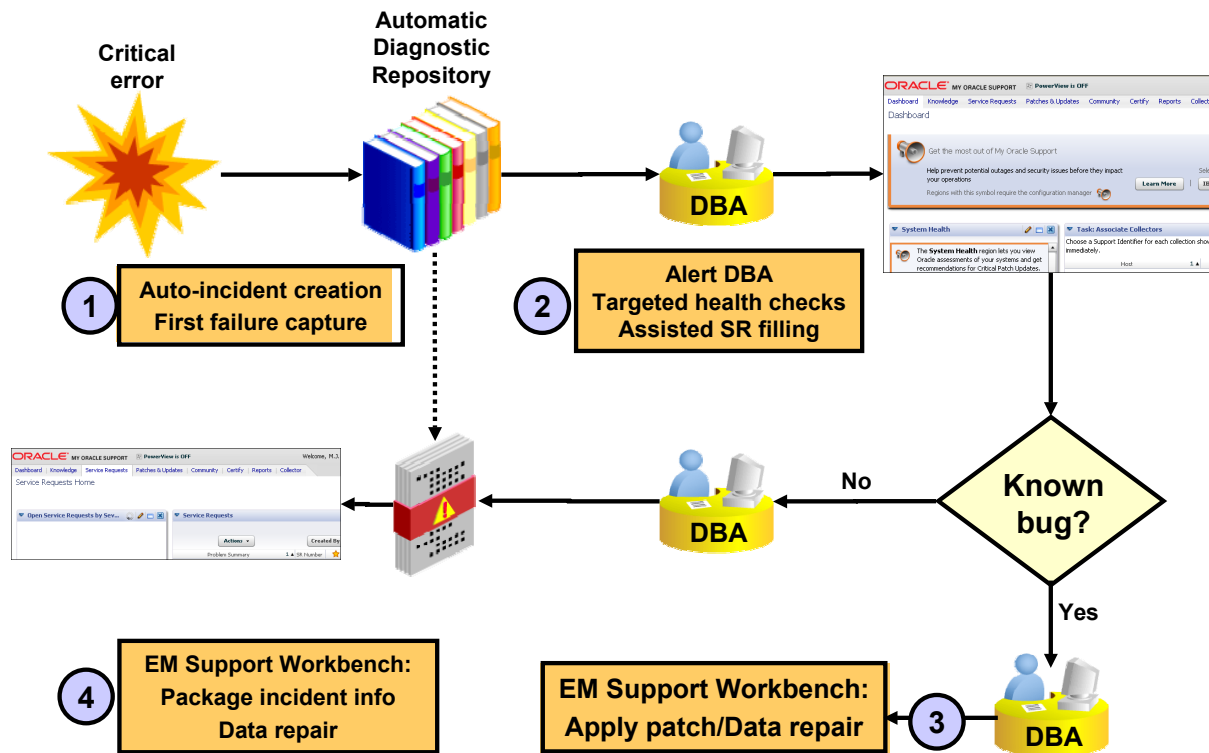
```
ORA-01578: ORACLE data block corrupted (file # 7, block # 3)
ORA-01110: data file 7: '/oracle/oradata/orcl/tools01.dbf'
ORA-01578: ORACLE data block corrupted (file # 2, block # 235)
ORA-01110: data file 2: '/oracle/oradata/orcl/undotbs01.dbf'
```

Once the blocks have been identified, run the RECOVER ... BLOCK command at the RMAN prompt, specifying the file and block numbers for the corrupted blocks.

```
RECOVER
DATAFILE 7 BLOCK 3
DATAFILE 2 BLOCK 235;
```

Automatic Diagnostic Workflow

Data Recovery Ad.
Block Corruption
> **ADR**
Health Monitor



Copyright © 2009, Oracle. All rights reserved.

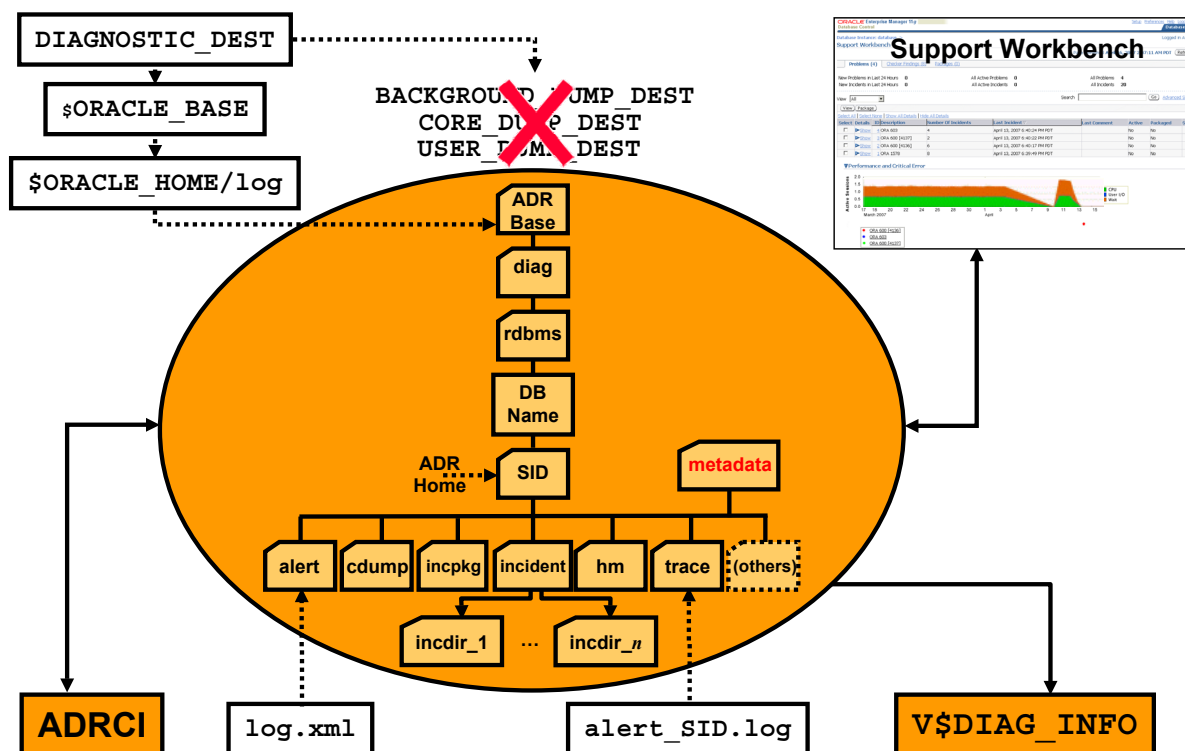
Automatic Diagnostic Workflow

An always-on, in-memory tracing facility enables database components to capture diagnostic data upon first failure for critical errors. A special repository, called Automatic Diagnostic Repository, is automatically maintained to hold diagnostic information about critical error events. This information can be used to create incident packages to be sent to Oracle Support Services for investigation.

Here is a typical workflow for a diagnostic session:

1. Incident causes an alert to be raised in Enterprise Manager (EM).
2. The DBA can view the alert via the EM Alert page.
3. The DBA can drill down to incident and problem details.
4. The DBA or Oracle Support Services can decide or ask for that information to be packaged and sent to Oracle Support Services via My Oracle Support. The DBA can add files to the data to be packaged automatically.

Automatic Diagnostic Repository



Copyright © 2009, Oracle. All rights reserved.

Automatic Diagnostic Repository (ADR)

The ADR is a file-based repository for database diagnostic data such as traces, incident dumps and packages, the alert log, Health Monitor reports, core dumps, and more. It has a unified directory structure across multiple instances and multiple products stored outside of any database. It is, therefore, available for problem diagnosis when the database is down.

Beginning with Oracle Database 11g R1, the database, Automatic Storage Management (ASM), Cluster Ready Services (CRS), and other Oracle products or components store all diagnostic data in the ADR. Each instance of each product stores diagnostic data underneath its own ADR home directory. For example, in a Real Application Clusters environment with shared storage and ASM, each database instance and each ASM instance have a home directory within the ADR. ADR's unified directory structure, consistent diagnostic data formats across products and instances, and a unified set of tools enable customers and Oracle Support to correlate and analyze diagnostic data across multiple instances.

The ADR root directory is known as the ADR base. Its location is set by the **DIAGNOSTIC_DEST** initialization parameter. If this parameter is omitted or left null, the database sets **DIAGNOSTIC_DEST** upon startup as follows: If environment variable **ORACLE_BASE** is set, **DIAGNOSTIC_DEST** is set to **\$ORACLE_BASE**. If environment variable **ORACLE_BASE** is not set, **DIAGNOSTIC_DEST** is set to **\$ORACLE_HOME/log**.

The ADR Command-Line Tool (ADRCI)

- ADRCI provides interaction with ADR from an operating system prompt.
- Using ADRCI, you can view diagnostic data within the Automatic Diagnostic Repository.

```
$ adrci
ADRCI: Release 11.1.0.5.0 - On Sat Jul 7 08:01:40 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.

ADR base = "/u01/app/oracle"

ADRCI> show incident
ADR Home = /u01/app/oracle/product/11.1.0/db_1/log/diag/rdbms/orcl/orcl:
*****
INCIDENT_ID PROBLEM_KEY CREATE_TIME
-----
1681          ORA-600_dbgris01:1,_addr=0xa9876541 17-JAN-07 09.17.44.843125...
1682          ORA-600_dbgris01:12,_addr=0xa9876542 18-JAN-07 09.18.59.434775...
2 incident info records fetched
```



Copyright © 2009, Oracle. All rights reserved.

The ADR Command-Line Tool (ADRCI)

ADRCI is a command-line tool that is part of the database fault diagnosability infrastructure. ADRCI enables you to:

- View diagnostic data within the Automatic Diagnostic Repository (ADR)
- Package incident and problem information into a zip file for transmission to Oracle Support

ADRCI can be used in interactive mode or within scripts. In addition, ADRCI can execute scripts of ADRCI commands in the same way that SQL*Plus executes scripts of SQL and PL/SQL commands. There is no need to log in to ADRCI, because the data in the ADR is not intended to be secure. ADR data is secured only by operating system permissions on the ADR directories.

The easiest way to package and otherwise manage diagnostic data is with the Support Workbench of Enterprise Manager (which assists with the resolution of database errors, as well as ASM errors).

ADRCI provides a command-line alternative to most of the functionality of Support Workbench, and adds capabilities such as listing and querying trace files. The example in the slide shows you an ADRCI session where you are listing all open incidents stored in ADR.

Note: For more information about ADRCI and the Support Workbench, refer to the *Oracle Database Utilities* guide.

The V\$DIAG_INFO View

```
SQL> SELECT * FROM V$DIAG_INFO;
```

NAME	VALUE
Diag Enabled	TRUE
ADR Base	/u01/app/oracle
ADR Home	/u01/app/oracle/diag/rdbms/orcl/orcl
Diag Trace	/u01/app/oracle/diag/rdbms/orcl/orcl/trace
Diag Alert	/u01/app/oracle/diag/rdbms/orcl/orcl/alert
Diag Incident	/u01/app/oracle/diag/rdbms/orcl/orcl/incident
Diag Cdump	/u01/app/oracle/diag/rdbms/orcl/orcl/cdump
Health Monitor	/u01/app/oracle/diag/rdbms/orcl/orcl/hm
Default Trace File	/u01/app/oracle/diag/.../trace/orcl_ora_11424.trc
Active Problem Count	3
Active Incident Count	8

ORACLE

Copyright © 2009, Oracle. All rights reserved.

The V\$DIAG_INFO View

The V\$DIAG_INFO view lists all important ADR locations:

- ADR Base: Path of the ADR base
- ADR Home: Path of the ADR home for the current database instance
- Diag Trace: Location of the text alert log and background/foreground process trace files
- Diag Alert: Location of an XML version of the alert log
- Diag Incident: Incident logs are written here.
- Diag Cdump: Diagnostic core files are written to this directory.
- Health Monitor: Location of logs from Health Monitor runs
- Default Trace File: Path to the trace file for your session. SQL trace files are written here.

Location for Diagnostic Traces

Diag Data	Previous Location	ADR Location
Foreground process traces	USER_DUMP_DEST	ADR_HOME/trace
Background process traces	BACKGROUND_DUMP_DEST	ADR_HOME/trace
Alert log data	BACKGROUND_DUMP_DEST	ADR_HOME/alert ADR_HOME/trace
Core dumps	CORE_DUMP_DEST	ADR_HOME/cdump
Incident dumps	USER BACKGROUND_DUMP_DEST	ADR_HOME/incident/incdir_ n

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Location for Diagnostic Traces

The table in the slide compares the different classes of trace data and dumps that reside both in Oracle Database 10g and Oracle Database 11g.

With Oracle Database 11g, there is no distinction between foreground and background traces files. Both types of files go into the *ADR_HOME/trace* directory.

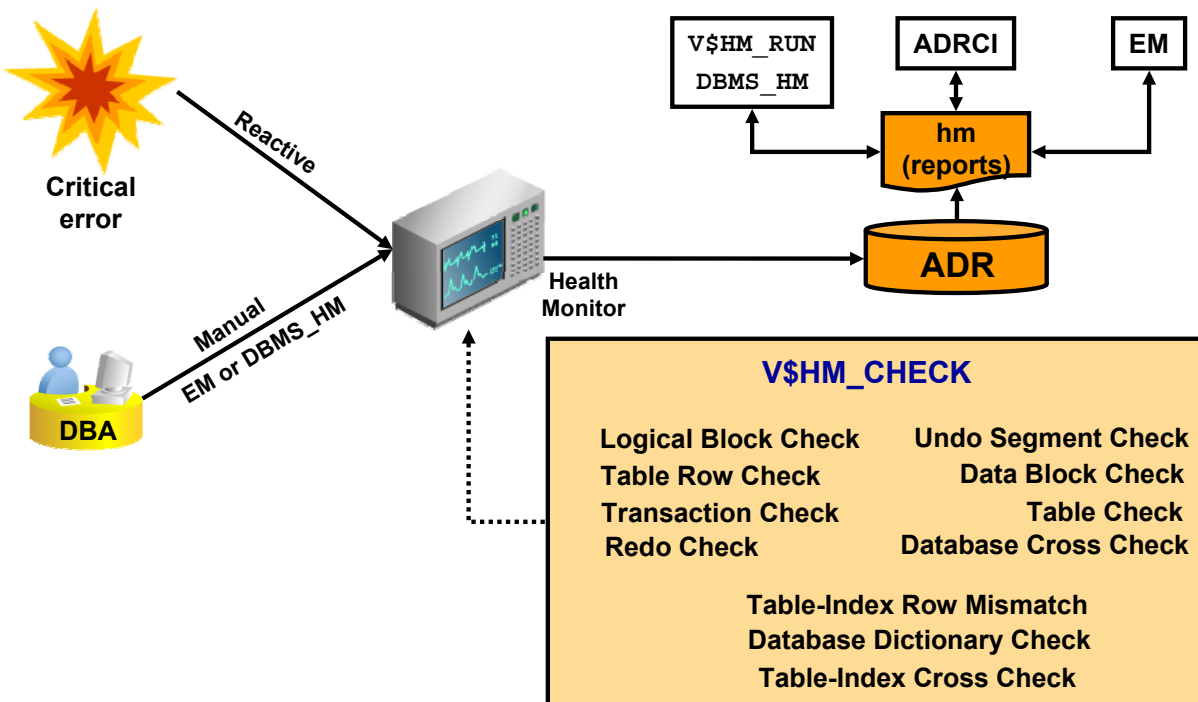
All nonincident traces are stored inside the *trace* subdirectory. This is the main difference compared with previous releases where critical error information is dumped into the corresponding process trace files instead of incident dumps. Incident dumps are placed in files separated from the normal process trace files starting with Oracle Database 11g.

The main difference between a trace and a dump is that a trace is more of a continuous output such as when SQL tracing is turned on, and a dump is a one-time output in response to an event such as an incident. Also, a core is a binary memory dump that is port specific.

Note: In the slide, *ADR_HOME* represents the path */u01/app/oracle/diag/rdbms/orcl/orcl*, assuming an instance name of *orcl*. However, there is no official environment variable called *ADR_HOME*.

Health Monitor: Overview

Data Recovery Ad.
Block Corruption
ADR
> [Health Monitor](#)



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Health Monitor: Overview

The Oracle database includes a framework called Health Monitor for running diagnostic checks on various components of the database. Health Monitor checks examine various components of the database, including files, memory, transaction integrity, metadata, and process usage. These checks generate reports of their findings as well as recommendations for resolving problems. The fault diagnosability infrastructure can run Health Monitor checks automatically in response to critical errors or the DBA, can manually run Health Monitor health checks by using either the DBMS_HM PL/SQL package or the Enterprise Manager interface.

For a complete description of all possible checks that Health Monitor can run, look at V\$HM_CHECK. These health checks fall into one of two categories:

- **DB-online:** These checks can be run while the database is open (that is, in OPEN mode).
- **DB-offline:** In addition to being “runnable” while the database is open, these checks can also be run when the instance is available and the database itself is closed (NOMOUNT mode).

After a checker has run, it generates a report containing information about the checker’s findings, including the priorities (low, high, or critical), descriptions of the findings and their consequences, and basic statistics about the execution. Health Monitor generates reports in XML and stores the reports in ADR. You can view these reports using either V\$HM_RUN, DBMS_HM, ADRCI, or Enterprise Manager.

Running Health Checks Manually: PL/SQL Example

```
SQL> exec dbms_hm.run_check('Database Dictionary Check',
                           'mycheck',0, 'TABLE_NAME=tab$');

SQL> set long 100000
SQL> select dbms_hm.get_run_report('mycheck') from dual;

DBMS_HM.GET_RUN_REPORT('mycheck')
-----
<?xml version="1.0" encoding="US-ASCII"?>
<HM-REPORT REPORT_ID="mycheck"><TITLE>HM Report: mycheck</TITLE>
  <RUN_INFO>
    <CHECK_NAME>Database Dictionary Check</CHECK_NAME>
    <RUN_ID>21</RUN_ID><RUN_NAME>mycheck</RUN_NAME>
    <RUN_MODE>MANUAL</RUN_MODE><RUN_STATUS>COMPLETED</RUN_STATUS> ...
  </RUN_INFO>
  <RUN_PARAMETERS><RUN_PARAMETER>TABLE_NAME=tab$</RUN_PARAMETER> ... </RUN_PARAMETERS>
  <RUN-FINDINGS><FINDING>
    <FINDING_NAME>Dictionary Inconsistency</FINDING_NAME><FINDING_ID>22</FINDING_ID>
    <FINDING_TYPE>FAILURE</FINDING_TYPE><FINDING_STATUS>OPEN</FINDING_STATUS>
    <FINDING_PRIORITY>CRITICAL</FINDING_PRIORITY> ...
    <FINDING_CREATION_TIME>...</FINDING_CREATION_TIME>
    <FINDING_MESSAGE>...invalid column number 7 on Object tab$ Failed</FINDING_MESSAGE>
    <FINDING_MESSAGE>Damaged ... Object SH.JFVTEST is referenced </FINDING_MESSAGE> ...
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Running Health Checks Manually: PL/SQL Example

You can use the `DBMS_HM.RUN_CHECK` procedure for running a health check. To call `RUN_CHECK`, supply the name of the check found in `V$HM_CHECK`, the name for the run (this is just a label used to retrieve reports later), and the corresponding set of input parameters for controlling its execution. You can view these parameters using the `V$HM_CHECK_PARAM`.

In the slide example, you want to run a Database Dictionary Check for `TAB$` table, considered an important core dictionary object. You call this run `MYCHECK`, and you do not want to set any timeout for this check.

When executed, you execute the `DBMS_HM.GET_RUN_REPORT` function to get the report extracted from `V$HM_RUN`, `V$HM_FINDING`, and `V$HM_RECOMMENDATION`. The output clearly shows you that a critical error was found in `TAB$`. This table contains an entry for a table with an invalid number of columns. Furthermore, the report gives you the name of the damaged table in `TAB$`.

When you call the `GET_RUN_REPORT` function, it generates the XML report file in the HM directory of your ADR. In the example, the file is called `HMREPORT_mycheck.hm`.

Note: Refer to the *Oracle Database PL/SQL Packages and Types Reference* for more information about `DBMS_HM`.

Viewing HM Reports Using the ADRCI Utility

```
adrci>>show hm_run
...
-----
RUN_ID                11081
RUN_NAME              HM_RUN_11081
CHECK_NAME            Database Cross Check
NAME_ID              2
MODE                 2
START_TIME            2007-04-13 03:20:31.161396 -07:00
RESUME_TIME
END_TIME              2007-04-13 03:20:37.903984 -07:00
MODIFIED_TIME         2007-04-17 01:16:37.106344 -07:00
TIMEOUT              0
FLAGS                0
STATUS               5
SRC_INCIDENT_ID       0
NUM_INCIDENTS         0
ERR_NUMBER            0
REPORT_FILE
...
adrci>>create report hm_run HM_RUN_11081
Adrci>>show report hm_run HM_RUN_11081
...
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Viewing HM Reports Using the ADRCI Utility

You can create and view Health Monitor checker reports using the ADRCI utility. To do that, ensure that operating system environment variables such as ORACLE_HOME are set properly, and then enter the following command at the operating system command prompt: `adrci`.

The utility starts and displays its prompt as shown in the slide. Optionally, you can change the current ADR home. Use the `SHOW HOMES` command to list all ADR homes, and the `SET HOMEPath` command to change the current ADR home.

You can then enter the `SHOW HM_RUN` command to list all the checker runs registered in the ADR repository and visible from `V$HM_RUN`. Locate the checker run for which you want to create a report and note the checker run name using the corresponding `RUN_NAME` field. The `REPORT_FILE` field contains a file name if a report already exists for this checker run. Otherwise, you can generate the report using the `CREATE REPORT HM_RUN` command as shown in the slide. To view the report, use the `SHOW REPORT HM_RUN` command.

Quiz

The Data Recovery Advisor handles both cases: when you cannot start up the database (because some required database files are missing, inconsistent, or corrupted) and when file corruptions are discovered during run time.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Quiz

After executing the `ADVISE FAILURE` command, the repair is automatically executed. So, it is no longer under your control.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Quiz

The ADR resides in the database. Therefore, an instance must be mounted for incident analysis.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Quiz

Which of the following checks can the Health Monitor perform?

1. Intuitive commit check
2. Memory check
3. Metadata check
4. Redo check
5. Transaction check
6. User alertness check
7. Undo segment check

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 3, 4, 5, 7

Summary

In this lesson, you should have learned how to:

- Detect and repair database corruption:
 - Use the new RMAN data repair commands to:
 - List failures
 - Receive a repair advice
 - Repair failures
 - Perform proactive failure checks
- Handle block corruption:
 - Verifying block integrity in real time
 - Performing block media recovery
- Set up Automatic Diagnostic Repository
- Run health checks

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 9 Overview: Diagnosing the Database

This practice covers the following topics:

- Discovering corruptions
- Repairing corruptions

ORACLE

Copyright © 2009, Oracle. All rights reserved.

10

Using Flashback Technology I

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

Flashback
> - Overview
- Query
- Table
- Transaction

After completing this lesson, you should be able to:

- Describe Flashback technology
- Perform Flashback Query
- Use Flashback Version Query
- Enable row movement on a table
- Perform Flashback Table operations
- Use Flashback Transaction Query
- Use Flashback Transaction

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Technology

Object Level	Scenario Examples	Flashback Technology	Depends On	Affects Data
Database	Truncate table; Undesired multitable changes made	Database	Flashback logs	TRUE
Table	Drop table	Drop	Recycle bin	TRUE
	Update with the wrong WHERE clause	Table	Undo data	TRUE
	Compare current data with data from the past	Query	Undo data	FALSE
	Compare versions of a row	Version	Undo data	FALSE
	Keep historical transaction data	Data Archive	Undo data	TRUE
Transaction	Investigate and back out suspect transactions	Transaction	Undo/redo from Archive logs	TRUE

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle University and Al-Khobara for Adaptive Knowledge use only

Flashback Technology

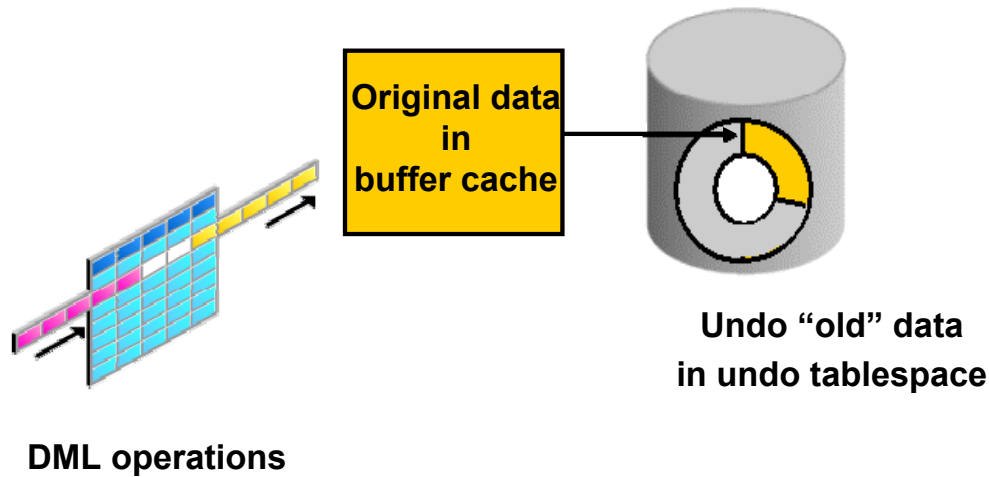
You can use Flashback technology when a logical corruption occurs in the Oracle database and you need to recover data quickly and easily. As with human errors, it is difficult to identify the objects and rows that are affected by an erroneous transaction. With Flashback technology, you can diagnose how errors are introduced into the database, and then repair the damage. You can view the transactions that have contributed to specific row modifications, view the entire set of versions of a given row during a specific time period, or just view data as it appeared at a specific time in the past. The table in the slide shows typical uses of Flashback technology. Flashback Database depends on the flashback logs to perform flashback. Flashback Drop uses the recycle bin. All other techniques use undo data.

Not all flashback features modify the database. Some are simply methods to query other versions of data; these are tools to investigate a problem and aid in recovery. The results of flashback queries help you do one of two things:

- Determine the type of database-modifying flashback operation to perform to fix the problem.
- Feed the result set of these queries into an INSERT, UPDATE, or DELETE statement that enables you to easily repair the erroneous data.

Flashback Data Archive enables you to use the preceding logical flashback features to access data from far back in the past.

Transactions and Undo



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Transactions and Undo

When a transaction starts, it is assigned to an undo segment. Throughout the life of the transaction, when data is changed, the original “old” values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the `V$TRANSACTION` view.

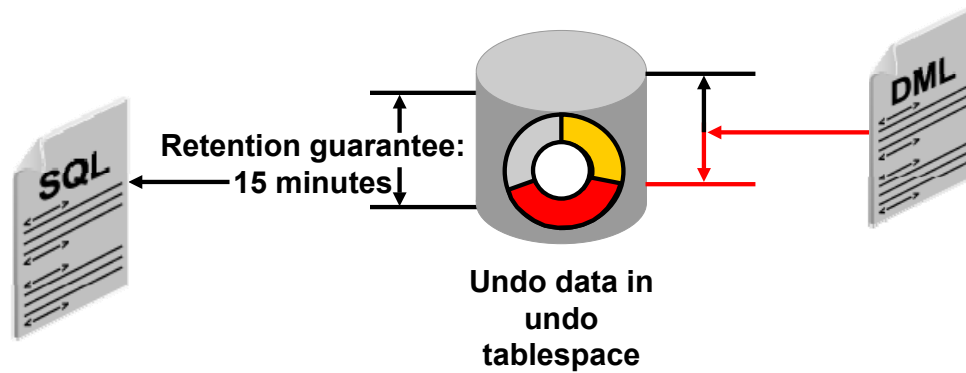
Undo segments are specialized segments that are automatically created by the instance as needed to support transactions. Like all segments, undo segments are made up of extents, which, in turn, consist of data blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions.

When transactions fill the blocks in their current undo segment extent, they are assigned another block in the same extent. If no free blocks remain in that extent, the transaction acquires a block from the next extent in the segment. If all extents are in use, the transaction either wraps around back into the first extent or requests that a new extent be allocated to the undo segment.

The diagram in the slide shows on the left a table icon with original data arriving from a DML operation. The original data is kept in the buffer cache (if not aged out) and then written to the undo tablespace (shown in circular form on the right).

Note: Parallel DML operations can actually cause a transaction to use more than one undo segment. To learn more about parallel DML execution, see the *Oracle Database Administrator's Guide*.

Guaranteeing Undo Retention



SELECT statements running 15 minutes or less are always satisfied.

A transaction that generates more undo than what there is space for will **fail.**

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Guaranteeing Undo Retention

The default undo behavior is to overwrite committed transactions that have not yet expired rather than to allow an active transaction to fail because of lack of undo space. In case of conflict, transactions have precedence over queries.

This behavior can be changed by guaranteeing retention. With guaranteed retention, undo retention settings are enforced even if they cause transactions to fail. (So in case of conflict, queries have precedence over transactions.)

RETENTION GUARANTEE is a tablespace attribute rather than an initialization parameter. This attribute can be changed using either SQL command-line statements or Enterprise Manager. The syntax to change an undo tablespace to guarantee retention is:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```

To return a guaranteed undo tablespace to its normal setting, use the following command:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;
```

You can set Undo Retention Guarantee in Enterprise Manager. Navigate to the Automatic Undo Management page. Click the current setting for Retention Guarantee (General/Undo Retention Settings) to modify it.

Preparing Your Database for Flashback

- Creating an undo tablespace
- Enabling Automatic Undo Management
- Specifying versus guaranteeing undo retention

Default database initialization parameters:

- `UNDO_MANAGEMENT= 'AUTO '`
- `UNDO_TABLESPACE= 'UNDOTBS1 '`
- `UNDO_RETENTION=900`

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Preparing Your Database for Flashback

To enable flashback features for an application, you must perform these tasks:

- Create an undo tablespace with enough space to keep the required data for flashback operations. The more often users update the data, the more space is required. The database administrator usually calculates the space requirement. If you are uncertain about your space requirements, you can start with an automatically extensible undo tablespace, observe it through one business cycle (for example, 1 or 2 days), collect undo block information with the `V$UNDO_STAT` view, calculate your space requirements, and use them to create an appropriately sized fixed undo tablespace. (The calculation formula is in the *Oracle Database Administrator's Guide*.)
- By default, Automatic Undo Management is enabled. If needed, enable Automatic Undo Management, as explained in the *Oracle Database Administrator's Guide*.
- For a fixed-size undo tablespace, the Oracle database automatically tunes the system to give the undo tablespace the best possible undo retention.
- For an automatically extensible undo tablespace (default), the Oracle database retains undo data to satisfy at a minimum, the retention periods needed by the longest-running query and the threshold of undo retention, specified by the `UNDO_RETENTION` parameter.

Preparing Your Database for Flashback (continued)

You can query `V$UNDOSTAT.TUNED_UNDORETENTION` to determine the amount of time for which undo is retained for the current undo tablespace. Setting the `UNDO_RETENTION` parameter does not guarantee, that unexpired undo data is not overwritten. If the system needs more space, the Oracle database can overwrite unexpired undo with more recently generated undo data.

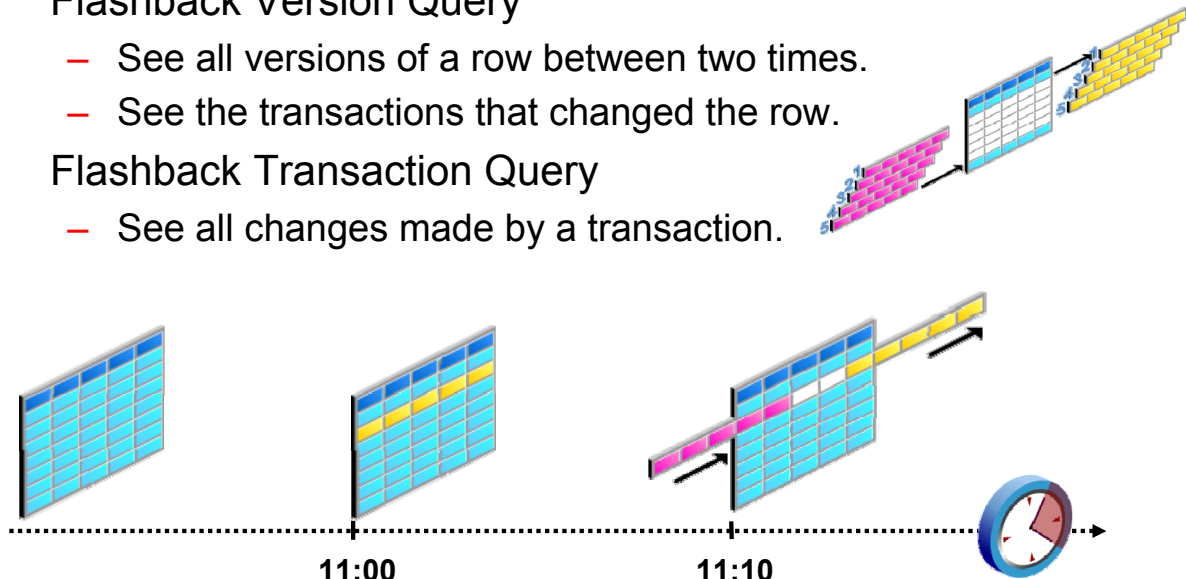
- Specify the `RETENTION GUARANTEE` clause for the undo tablespace to ensure that unexpired undo data is not discarded.
- Grant flashback privileges to users, roles, or applications that need to use flashback features.

To satisfy long retention requirements, create a Flashback Data Archive.

Using Flashback Technology to Query Data

Flashback
- Overview
> - Query
- Table
- Transaction

- Flashback Query
 - Query all data at a specified point in time.
- Flashback Version Query
 - See all versions of a row between two times.
 - See the transactions that changed the row.
- Flashback Transaction Query
 - See all changes made by a transaction.



Copyright © 2009, Oracle. All rights reserved.

ORACLE

Using Flashback Technology to Query Data

Flashback technology provides the capability to query past versions of schema objects, query historical data, and perform change analysis. Every transaction logically generates a new version of the database. With Flashback technology, you can navigate through these versions to find an error and its cause:

- **Flashback Query:** Query all data as it existed at a specific point in time.
- **Flashback Version Query:** See all versions of rows between two times and the transactions that changed the row.
- **Flashback Transaction Query:** See all changes made by a transaction and, if needed, roll back a transaction with “undo” SQL commands.

Flashback Query

Use to query all data at a specified point in time.



```
SELECT employee_id, salary FROM employees
AS OF TIMESTAMP <T1>
WHERE employee_id = 200
```

ORACLE

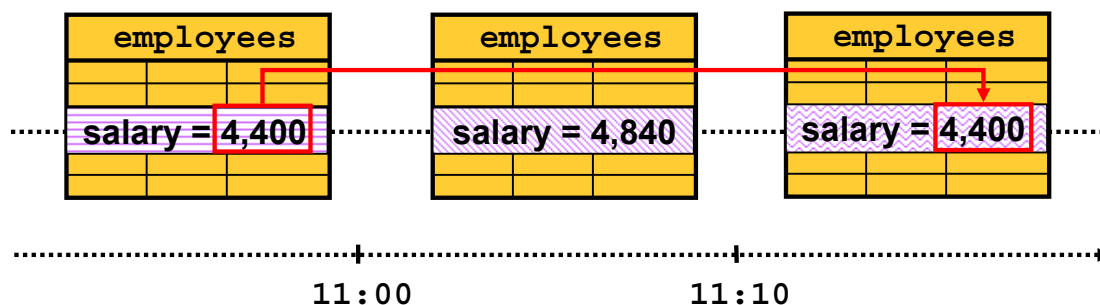
Copyright © 2009, Oracle. All rights reserved.

Flashback Query

With the Flashback Query feature, you can perform queries as of a certain time. By using the AS OF clause of the SELECT statement, you can specify the time stamp for which to view the data. This is useful for analyzing a data discrepancy.

Note: TIMESTAMP and SCN are valid options for the AS OF clause.

Flashback Query: Example



```
UPDATE employees
SET salary =
  (SELECT salary FROM employees
   AS OF TIMESTAMP TO_TIMESTAMP
    ('2005-05-04 11:00:00', 'yyyy-mm-dd hh24:mi:ss')
   WHERE employee_id = 200)
WHERE employee_id = 200
```

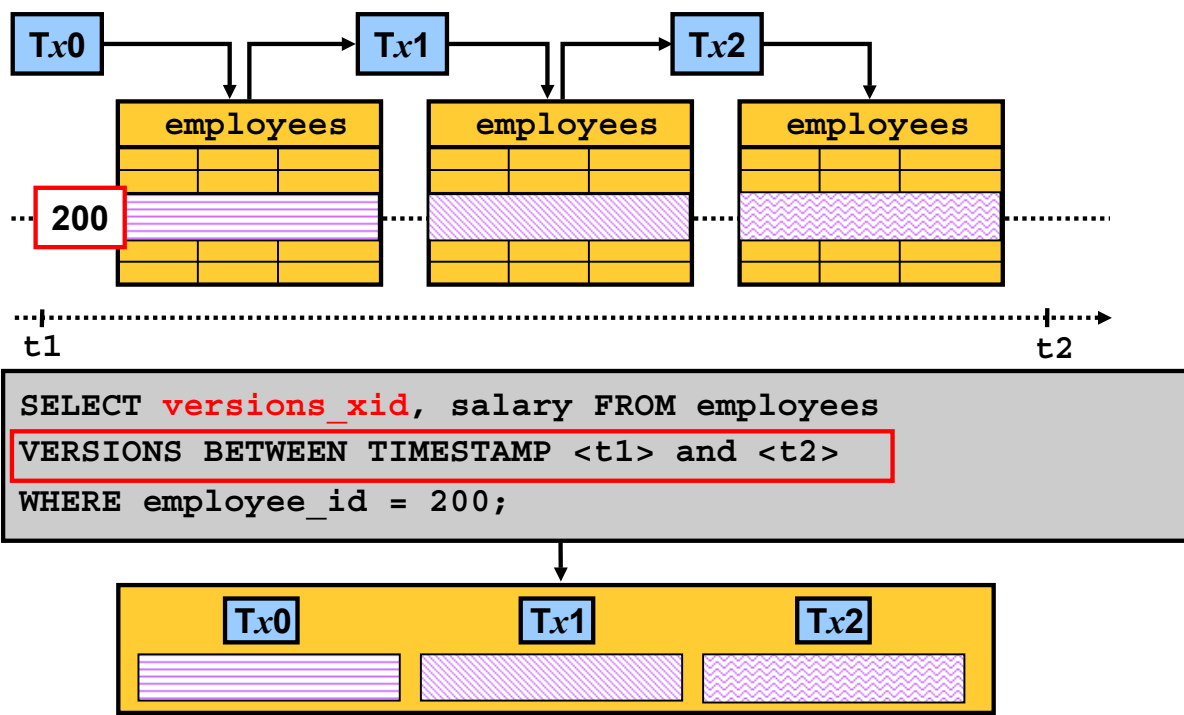
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Query: Example

If a raise has been erroneously given to a particular employee recently, you can update the salary again, assigning the salary provided by a subquery that returns the flashed-back value.

Flashback Version Query



Copyright © 2009, Oracle. All rights reserved.

Flashback Version Query

With Flashback Query, you can perform queries on the database as of a certain time span or range of user-specified system change numbers (SCNs). The Flashback Version Query feature enables you to use the `VERSIONS` clause to retrieve all the versions of the rows that exist between two points in time or two SCNs.

The rows returned by Flashback Version Query represent a history of changes for the rows across transactions. Flashback Version Query retrieves only committed occurrences of the rows. Uncommitted row versions within a transaction are not shown. The rows returned also include deleted and subsequently reinserted versions of the rows.

You can use Flashback Version Query to retrieve row history. It provides you with a way to audit the rows of a table and retrieve information about the transactions that affected the rows. You can then use the returned transaction identifier to perform transaction mining by using LogMiner or to perform a Flashback Transaction Query, as described later in this lesson.

Note: `VERSIONS_XID` is a pseudocolumn that returns the transaction identifier of the corresponding version of a row.

Flashback Version Query: Considerations

- The `VERSIONS` clause cannot be used to query:
 - External tables
 - Temporary tables
 - Fixed tables
 - Views
- The `VERSIONS` clause cannot span DDL commands.
- Segment shrink operations are filtered out.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Version Query: Considerations

The `VERSIONS` clause cannot be used to query the following types of tables:

- External tables
- Temporary tables
- Fixed tables

You cannot use the `VERSIONS` clause to query a view. However, a view definition can use the `VERSIONS` clause.

The `VERSIONS` clause in a `SELECT` statement cannot produce versions of rows across the DDL statements that change the structure of the corresponding tables. This means that the query stops producing rows after it reaches a time in the past when the table structure was changed.

Certain maintenance operations, such as a segment shrink, may move table rows across blocks. In this case, the version query filters out such phantom versions because the row data remains the same.

Quiz

Flashback Query compares current data with data from the past. To do so, it uses both undo and redo data.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Quiz

Select the correct statement:

1. Flashback Version Query uses undo data and modifies data.
2. Flashback Version Query uses undo data and does not modify data.
3. Flashback Version Query uses both undo and redo data.

ORACLE

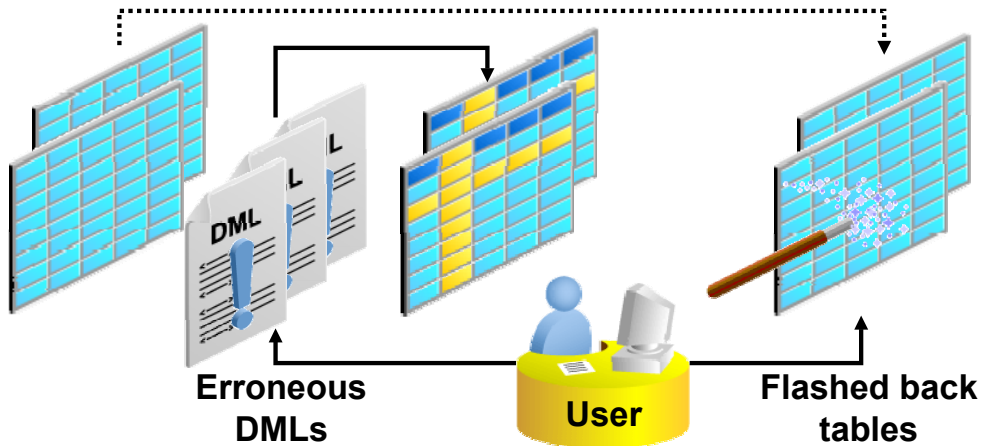
Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Flashback Table: Overview

Flashback
- Overview
- Query
> - Table
- Transaction

- Flashback Table recovers tables to a specific point in time.
- Flashback Table is an in-place operation.
- The database stays online.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Table Overview

Using Flashback Table, you can recover a set of tables to a specific point in time without having to perform traditional point-in-time recovery operations.

A Flashback Table operation is done in-place, while the database is online, by rolling back only the changes that are made to the given tables and their dependent objects.

A Flashback Table statement is executed as a single transaction. All tables must be flashed back successfully, or the entire transaction is rolled back.

Note: You can use Flashback Versions Query and Flashback Transaction Query to determine the appropriate flashback time.

Flashback Table

- Using Flashback Table, you can recover a table or tables to a specific point in time without restoring a backup.
- Data is retrieved from the undo tablespace to perform a Flashback Table operation.
- You require the `FLASHBACK ANY TABLE` or the `FLASHBACK` object privilege on the specific table.
- `SELECT`, `INSERT`, `DELETE`, and `ALTER` privileges on the table to be flashed back are required.
- You *must* enable row movement on the table that you are performing the flashback operation on.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Table

With Flashback Table, you can recover a table or tables to a specific point in time without restoring a backup. When you use this feature, the data in tables and their associated objects (indexes, constraints, triggers, and so on) is restored. The data used to satisfy a Flashback Table request is retrieved from the undo tablespace. You can use Flashback Versions Query and Flashback Transaction Query to determine the appropriate flashback time.

Flashback Table provides a way for users to easily and quickly recover from accidental modifications without a database administrator's involvement. You must grant the `FLASHBACK TABLE` or `FLASHBACK ANY TABLE` system privilege to any user that uses the Flashback Table feature. In addition, you must grant the `SELECT`, `INSERT`, `DELETE`, and `ALTER` object privileges to the user.

You can use Enterprise Manager to flash back a table. The wizard guides you through the process.

Note: Enabling row movement is described on the next page.

Enabling Row Movement on a Table

Edit Table: HR.EMPLOYEES

Actions:

[General](#) [Constraints](#) [Segments](#) [Storage](#) **[Options](#)** [Statistics](#) [Indexes](#)

Enable Row Movement

☐ Parallel - Use multiple threads when creating this object or when executing DML against this object.
Parallel Degree ☒ Default ☐ Value

☐ Cache - Place frequently accessed data to the top of the buffer cache.

[General](#) [Constraints](#) [Segments](#) [Storage](#) **[Options](#)** [Statistics](#) [Indexes](#)

```
ALTER TABLE employees ENABLE ROW MOVEMENT;
```

Enabling Row Movement on a Table

You must enable row movement on a table to be able to flash back the table. When you enable row movement, the Oracle server can move a row in the table.

Using Enterprise Manager, you can enable row movement on a table by performing the following steps:

1. Select Tables in the Database Objects region of the Schema property page. Enter the schema name to search for the table, and click Go.
2. Click the table name of the table for which you want to enable row movement. You are now on the View Table page.
3. Click Edit, which takes you to the Edit Table page.
4. Click the Options tab, where you can change the Enable Row Movement setting for the table.
5. Set Enable Row Movement to Yes, and click Apply. The update confirmation message is displayed.

Performing Flashback Table

Perform Object Level Recovery: Point-in-time

Recovery Scope **Tables** Cancel Step 1 of 7 Next

Operation Type **Flashback Existing Tables**

Specify the point in time to which to recover.

☒ Evaluate row changes and transactions to decide on a point in time

* Table Example: SCOTT.EMP

☐ Flashback to a timestamp

Date Time : ☒ AM ☐ PM

Example: Mar 19, 2003

☐ Flashback to a restore point

Restore Point

☐ Flashback to a known SCN

SCN

Perform Recovery

Oracle Advised Recovery

Oracle did not detect any failures. Advise and Recover

User Directed Recovery

Recovery Scope Recover

Operation Type ☒ Flashback Existing Tables ☐ Flashback Dropped Tables

FLASHBACK TABLE hr.departments TO TIMESTAMP TO_TIMESTAMP('2007-04-05 21:00:00', 'YYYY-MM-DD HH24:MI:SS');

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performing Flashback Table

You can use Enterprise Manager to flash back a table by performing the following steps:

1. Select Perform Recovery in the Backup/Recovery region on the Availability property page.
2. In the Object Level Recovery region, select Tables from the Object Type drop-down list.
3. Select Flashback Existing Tables as Operation Type. Click Recover. The “Perform Object Level Recovery: Point-in-time” page is displayed.
4. Select “Flashback to a timestamp” or “Flashback to a known SCN” and then specify a time stamp or SCN to flash back to, and click Next.
5. Click Add Tables to add tables to the list for the flashback operation. Click Next.
6. The Dependency Options page appears if there are dependent tables. Select the desired option for dealing with dependent tables. Typically, you would select “Cascade” to ensure a consistent flashback. Click Next.
7. The “Perform Object Level Recovery: Review” page appears. Review the information and click Submit. The Confirmation page appears.

Note: You can also flash back tables from the Tables link in the Schema region of the Administration page.

Flashback Table: Considerations

- The FLASHBACK TABLE command executes as a single transaction, acquiring exclusive DML locks.
- Statistics are not flashed back.
- Current indexes and dependent objects are maintained.
- Flashback Table operations:
 - Cannot be performed on system tables
 - Cannot span DDL operations
 - Generate undo and redo data

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Table: Considerations

- The entire FLASHBACK TABLE statement is executed within a single transaction. All or none of the specified tables are flashed back.
- Flashback Table acquires exclusive data manipulation language (DML) locks on all the tables that are specified in the statement over the period of time when the operation is in progress.
- Statistics of impacted objects are not flashed back.
- All existing indexes are maintained. Dropped indexes are not re-created. Dependent on-commit materialized views are also maintained automatically.
- Tables specified in the FLASHBACK TABLE statement are flashed back, provided that none of the table constraints are violated. If any constraints are violated during flashback execution, the operation is aborted and the tables are left in the same state as they were just before the FLASHBACK TABLE statement invocation.
- You cannot perform Flashback Table to a particular time that is older than the time of the execution of a data definition language (DDL) operation that altered the structure of or shrunk a table that would be involved in the flashback operation. This restriction does not apply to DDL statements that only change storage attributes of the tables.
- Flashback Table cannot be performed on system tables, remote tables, and fixed tables.

Quiz

Select all correct statements:

1. The database can remain open when a table is flashed back.
2. Flashback Table is executed as a single transaction.
3. Flashback Table requires backups to be available.
4. Flashback Table is based on undo data.

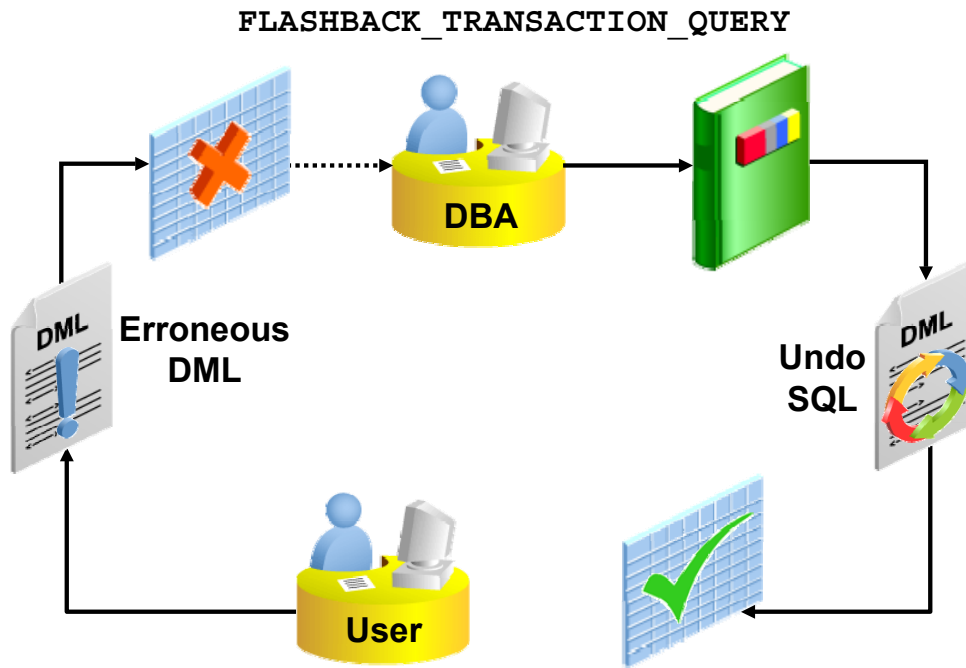
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 2, 4

Flashback Transaction Query

Flashback
- Overview
- Query
- Table
> - Transaction



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Transaction Query

Flashback Transaction Query is a diagnostic tool that you can use to view changes made to the database at the transaction level. This enables you to diagnose problems in your database and perform analysis and audits of transactions.

You can use the `FLASHBACK_TRANSACTION_QUERY` view to determine all the necessary SQL statements that can be used to undo the changes made either by a specific transaction or during a specific period of time.

Using Enterprise Manager to Perform Flashback Transaction Query

Select	Transaction ID	DB User	Commit Timestamp	Redo Records	Transaction Summary - Updates (upd), Inserts (ins), Deletes (del), Other (oth)
	06001600C2030000	HR	Aug 20, 2009 4:17:03 PM	5	HR. REGIONS (5 ins)
	08001C0052030000	HR	Aug 20, 2009 4:17:05 PM	2	HR. REGIONS (2 upd)
	04001000E0300000	HR	Aug 20, 2009 4:17:06 PM	3	HR. REGIONS (3 upd)
	09001F00A0030000	HR	Aug 20, 2009 4:17:07 PM	1	HR. REGIONS (1 upd)

Transaction ID 08001C0052030000		Start SCN 1106834		Start Time Aug 20, 2009 4:17:05 PM	
DB User UNKNOWN		Commit SCN 1106835		Commit Time Aug 20, 2009 4:17:05 PM	
OS User		Machine Name			
SCN ▲	Operation	Schema	Table	SQL Redo	
1106834	START			set transaction read write;	
1106834	UPDATE	HR	REGIONS	update "HR"."REGIONS" set "REGION_NAME" = 'Two Poles' where "REGION_ID" = 10 and "REGION_NAME" = 'Pole' and ROWID = 'AAAR5JAAFAAAACNAAA';	
1106834	UPDATE	HR	REGIONS	update "HR"."REGIONS" set "REGION_NAME" = 'Many Moons' where "REGION_ID" = 20 and "REGION_NAME" = 'Moon' and ROWID = 'AAAR5JAAFAAAACNAAB';	
1106835	COMMIT			commit;	

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Enterprise Manager to Perform Flashback Transaction Query

This feature is used in conjunction with the Flashback Version Query feature with the help of the Perform Recovery Wizard. On the “Perform Object Level Recovery: Choose SCN” page, click the corresponding Transaction ID link in the Flashback Version Query Result region.

In the example in the slide, a Flashback Version Query is performed on the JOBS table to retrieve the three versions of the JOBS row for JOB_ID = 'AD_PRES'. Then, one of the transaction IDs is clicked, showing all the changes that were part of that transaction. Notice that in addition to the JOBS table update, there was also an update to the EMPLOYEES table in that transaction.

Flashback Transaction Query: Considerations

- DDL commands are seen as dictionary updates.
- Flashback Transaction Query on a transaction underlying a DDL command displays the data dictionary changes.
- Dropped objects appear as object numbers.
- Dropped users appear as user identifiers.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Transaction Query: Considerations

Within the database, DDL operations are nothing but a series of space management operations and changes to the data dictionary. Flashback Transaction Query on a transaction underlying a DDL command displays the changes made to the data dictionary.

When Flashback Transaction Query involves tables that have been dropped from the database, the table names are not reflected. Instead, object numbers are used.

If the user who executed a transaction is dropped, Flashback Transaction Query of that transaction displays the corresponding user ID only, and not the username.

Note: When there is not enough undo data for a specific transaction, a row with a value of UNKNOWN in the OPERATION column of FLASHBACK_TRANSACTION_QUERY is returned.

Flashback Transaction

- Setting up Flashback Transaction prerequisites
- Stepping through a possible workflow
- Using the Flashback Transaction Wizard
- Querying transactions with and without dependencies
- Choosing back-out options and flashing back transactions
- Reviewing the results

ORACLE

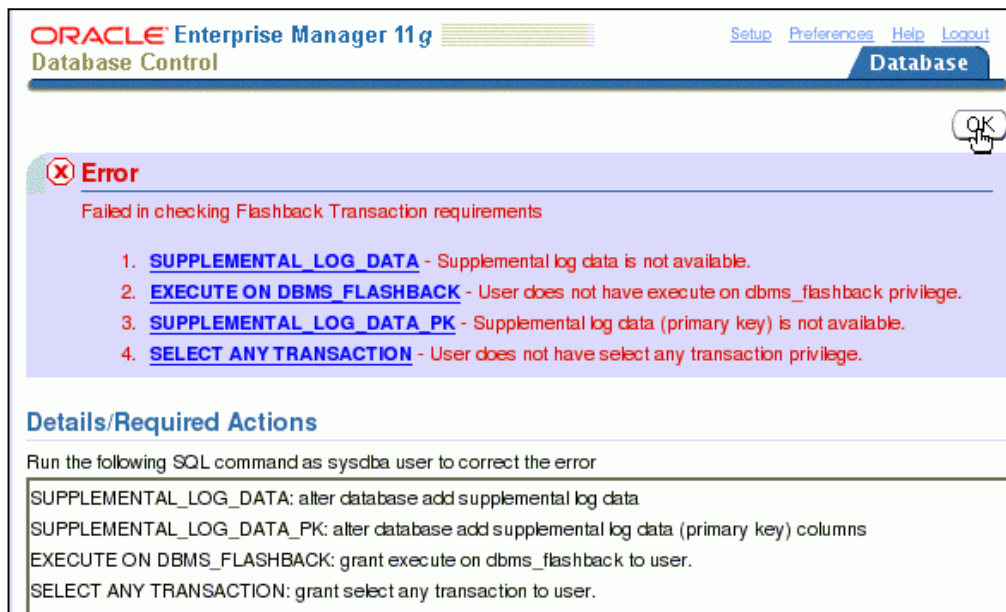
Copyright © 2009, Oracle. All rights reserved.

Flashback Transaction

With Flashback Transaction, you can reverse a transaction and dependant transactions. Oracle Database determines the dependencies between transactions and, in effect, creates a compensating transaction that reverses the unwanted changes. The database rewinds to a state as if the transaction, and any transactions that could be dependent on it, never occurred.

You can use the Flashback Transaction functionality from within Enterprise Manager or with PL/SQL packages.

Prerequisites



Prerequisites

In order to use this functionality, supplemental logging must be enabled and the correct privileges established. For example, the HR user in the HR schema decides to use Flashback Transaction for the REGIONS table. The SYSDBA performs the following setup steps in SQL*Plus:

```
alter database add supplemental log data;  
alter database add supplemental log data (primary key) columns;  
grant execute on dbms_flashback to hr;  
grant select any transaction to hr;
```

Flashing Back a Transaction

- You can flash back a transaction with Enterprise Manager or from the command line.
- EM uses the Flashback Transaction Wizard, which calls the `DBMS_FLASHBACK.TRANSACTION_BACKOUT` procedure with the `NOCASCADE` option.
- If the PL/SQL call finishes successfully, it means that the transaction does not have any dependencies and a single transaction is backed out successfully.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashing Back a Transaction

Security privileges

To flash back or back-out a transaction—that is, to create a compensating transaction—you must have the `SELECT`, `FLASHBACK`, and `DML` privileges on all affected tables.

Conditions of Use

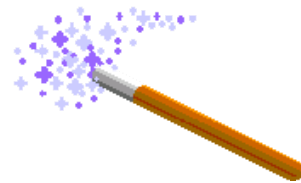
- Transaction back-out is not supported across conflicting DDL.
- Transaction back-out inherits data type support from LogMiner. See the Oracle Database 11g documentation for supported data types.

Recommendation

- When you discover the need for transaction back-out, performance is better if you start the back-out operation sooner. Large redo logs and high transaction rates result in slower transaction back-out operations.
- Provide a transaction name for the back-out operation to facilitate later auditing. If you do not provide a transaction name, it will be automatically generated for you.

Possible Workflow

1. Viewing data in a table
2. Discovering a logical problem
3. Using Flashback Transaction
 1. Performing a query
 2. Selecting a transaction
 3. Flashing back a transaction (with no conflicts)
 4. Choosing other back-out options (if conflicts exists)
4. Reviewing Flashback Transaction results



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Possible Workflow

Assume that several transactions occurred as indicated below:

```
connect hr
```

```
Enter password: oracle_4U <<< not displayed
```

```
INSERT INTO hr.regions VALUES (5,'Pole');
```

```
COMMIT;
```

```
UPDATE hr.regions SET region_name='Poles' WHERE region_id = 5;
```

```
UPDATE hr.regions SET region_name='North and South Poles' WHERE region_id  
= 5;
```

```
COMMIT;
```

```
INSERT INTO hr.countries VALUES ('TT','Test Country',5);
```

```
COMMIT;
```

```
connect sys/<password> as sysdba
```

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Flashback Transaction Wizard

The screenshot shows the 'Flashback Transaction: Perform Query' step of the wizard. At the top, a progress bar indicates four steps: 'Perform Query' (selected), 'Select Transaction', 'Show Dependencies', and 'Review'. Below the progress bar, the title 'Flashback Transaction: Perform Query' is displayed. The 'Database' is set to 'orcl' and the 'Operation Type' is 'Flashback Transaction'. There are 'Cancel' and 'Next' buttons, with 'Step 1 of 4' indicating the current step. A text box explains: 'Specify the time range to begin querying. The time range is initialized to the last hour but any available (on disk) archived logs can be queried. You may need to specify additional columns (or reduce the time range) to further narrow the results.' Under 'Query Time Range', 'Time Range' is selected over 'SCN Range'. The 'Start Time' is 'Aug 20, 2009' at '05:19 PM', and the 'End Time' is 'Aug 20, 2009' at '05:33 PM'. Two tips are shown: 'TIP The oldest time available on disk is Aug 20, 2009 5:19:26 PM' with a link to 'View Archived Logs', and 'TIP Earlier start times are available by restoring archived logs'. Under 'Query Filter', the 'Table' is 'HR.REGIONS' with a magnifying glass icon and examples: 'Scott.Emp, Scott.%'.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Transaction Wizard

In Enterprise Manager, select `HR.REGIONS` under Table, select Flashback Transaction from the Actions drop-down list, and then click Go. This invokes the Flashback Transaction Wizard for your selected table. The Flashback Transaction: Perform Query page is displayed.

Select the appropriate time range and add query parameters. (The more specific you can be, the shorter is the search of the Flashback Transaction Wizard.)

Without Enterprise Manager, use the `DBMS_FLASHBACK.TRANSACTION_BACKOUT` procedure, which is described in the *PL/SQL Packages and Types Reference*. Essentially, you take an array of transaction IDs as the starting point of your dependency search. For example:

```
CREATE TYPE XID_ARRAY AS VARRAY(100) OF RAW(8);
CREATE OR REPLACE PROCEDURE TRANSACTION_BACKOUT(
  numberOfXIDs NUMBER, -- number of transactions passed as input
  xids XID_ARRAY, -- the list of transaction ids
  options NUMBER default NOCASCADE, -- back out dependent
  txn timeHint TIMESTAMP default MINTIME -- time hint on the txn
  start
);
```


Choosing Other Back-out Options

Flashback Transaction: Show Dependencies

Database: **orcl** Cancel Back Step 3 of 4 Next

Operation Type: **Flashback Transaction**

Flashback transaction is performed with 'Nonconflict only' recovery option. Only non-conflicting changes of the target transaction will be backed out. You can further examine the details of dependent transactions and change the recovery option if needed.

Change Recovery Option

Transaction ID	Transaction Property
0100040073050000	Normal
0700170064050000	Normal
0200160059070000	Normal
0700170064050000	Normal
0700170064050000	Normal

☒ **TIP** Transaction property indicates if this is a normal or a compensating transaction.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Choosing Other Back-out Options

The TRANSACTION_BACKOUT procedure checks dependencies, such as:

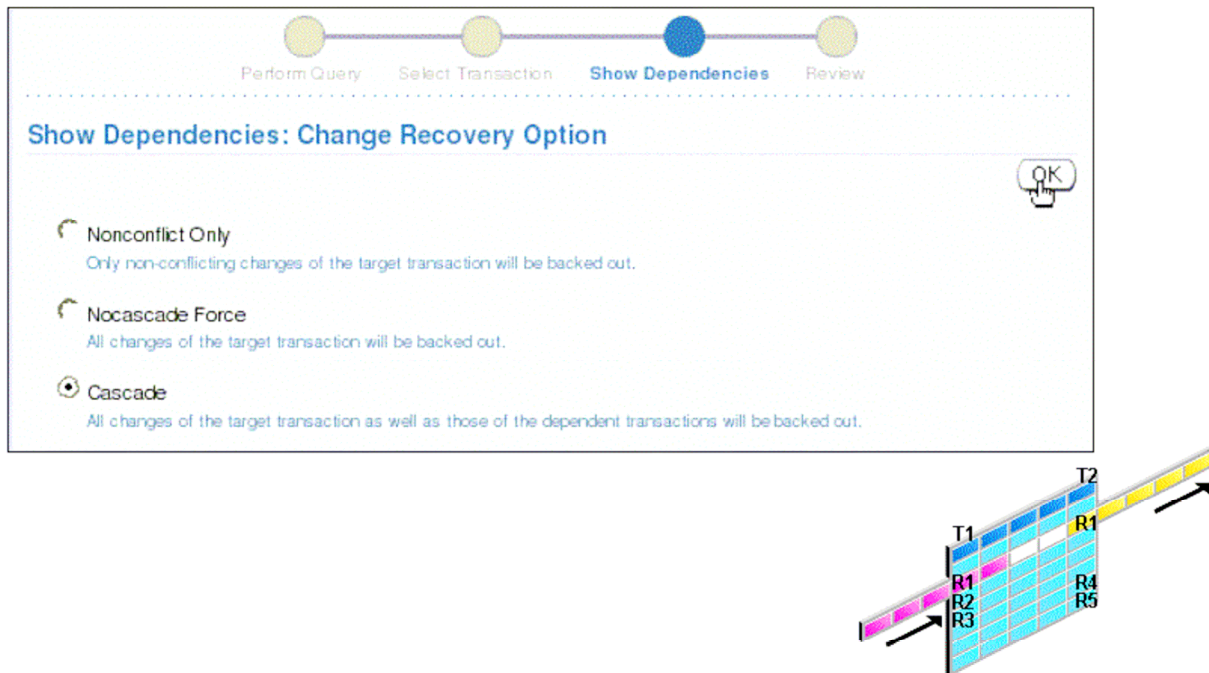
- Write-after-write (WAW)
- Primary and unique constraints
- Foreign key constraints

A transaction can have a WAW dependency, which means a transaction updates or deletes a row that has been inserted or updated by a dependent transaction. This can occur, for example, in a master/detail relationship of primary (or unique) and mandatory foreign key constraints.

To understand the difference between the NONCONFLICT_ONLY and the NOCASCADE_FORCE options, assume that the T1 transaction changes rows R1, R2, and R3 and the T2 transaction changes rows R1, R3, and R4. In this scenario, both transactions update row R1, so it is a “conflicting” row. The T2 transaction has a WAW dependency on the T1 transaction. With the NONCONFLICT_ONLY option, R2 and R3 are backed out, because there is no conflict and it is assumed that you know best, what to do with the R1 row. With the NOCASCADE_FORCE option, all three rows (R1, R2, and R3) are backed out.

Note: This screenshot is not part of the workflow example, but shows additional details of a more complex situation.

Choosing Other Back-out Options



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Choosing Other Back-out Options (continued)

The Flashback Transaction Wizard works as follows:

If the `DBMS_FLASHBACK.TRANSACTION_BACKOUT` procedure with the `NOCASCADE` option fails (because there are dependent transactions), you can change the recovery options.

- With the Nonconflict Only option, nonconflicting rows within a transaction are backed out, which implies that database consistency is maintained (although the transaction atomicity is broken for the sake of data repair).
- If you want to forcibly back out the given transactions, without paying attention to the dependent transactions, use the Nocascade Force option. The server simply executes the compensating DML commands for the given transactions in reverse order of their commit times. If no constraints break, you can proceed to commit the changes, or else roll back.
- To initiate the complete removal of the given transactions and all their dependents in a post-order fashion, use the Cascade option.

Note: This screenshot is not part of the workflow example, but shows additional details of a more complex situation.

Final Steps Without EM

After choosing your back-out option, the dependency report is generated in the `DBA_FLASHBACK_TXN_STATE` and `DBA_FLASHBACK_TXN_REPORT` views.

- Review the dependency report that shows all transactions which were backed out.
- Commit the changes to make them permanent.
- Roll back to discard the changes.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Final Steps Without EM

The `DBA_FLASHBACK_TXN_STATE` view contains the current state of a transaction: whether it is alive in the system or effectively backed out. This table is atomically maintained with the compensating transaction. For each compensating transaction, there could be multiple rows, where each row provides the dependency relationship between the transactions that have been compensated by the compensating transaction.

The `DBA_FLASHBACK_TXN_REPORT` view provides detailed information about all compensating transactions that have been committed in the database. Each row in this view is associated with one compensating transaction.

For a detailed description of these tables, see the *Oracle Database Reference*.

Quiz

You discover that Jim's salary was updated twice. The first update was correct, but the second was done by mistake. Before you discovered this problem, other rows in the `EMPLOYEES` table are updated correctly. Which technology should you use to fix this error?

1. Flashback Database
2. Flashback Query
3. Flashback Transaction

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 3

Summary

In this lesson, you should have learned how to:

- Describe Flashback technology
- Perform Flashback Query
- Use Flashback Version Query
- Enable row movement on a table
- Perform Flashback Table operations
- Use Flashback Transaction Query
- Use Flashback Transaction

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 10 Overview: Performing Flashback Transaction Backout

This practice covers the following topics:

- Querying a transaction
- Performing Flashback Transaction backout

ORACLE

Copyright © 2009, Oracle. All rights reserved.