
Enterprise DBA Part 1B: Backup and Recovery Workshop

Volume 1 • Instructor Guide

30050GC10

Production 1.0

August 1999

M09097

ORACLE®

Authors

Dominique Jeunot
Shankar Raman

Technical Contributors and Reviewers

David Austin
Ben van Balen
Ralf Durben
Bruce A. Ernst
Joel Goodman
Bert van Gorkom
Scott Gossett
Lex de Haan
John Hough Jr.
Alexander Hunold
Peter Kilpatrick
Stefan Lindblad
Hanne Rue Rasmussen
Robert Thome
Jean-Francois Verrier
Steven Wertheimer

Publisher

John B Dawson

Copyright © Oracle Corporation, 1999. All rights reserved.

This documentation contains proprietary information of Oracle Corporation. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c) (1) (ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of the Worldwide Education Services group of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with "Restricted Right," as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box 659806, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

SQL*Loader, SQL*Net, SQL*Plus, Net8, Oracle Call Interface, Oracle7, Oracle8, Developer/2000, Developer/2000 Forms, Designer/2000, Oracle Enterprise Manager, Oracle Parallel Server, Oracle Server Manager, PL/SQL, Pro*C, Pro*C/C++, and Trusted Oracle are trademarks or registered trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

Contents

Preface

Profile	xvii
Related Publications	xix
Typographic Conventions	xx

Introduction

Objectives	I-2
------------	-----

Lesson 1: Backup and Recovery Considerations

Objectives	1-2
Overview	1-4
Defining a Backup and Recovery Strategy	1-5
Business Requirements	1-6
Operational Requirements	1-7
Technical Requirements	1-9
Disaster Recovery Issues	1-11
Oracle Availability and Features	1-13
Summary	1-14

Lesson 2: Oracle Recovery Structures and Processes

Objectives	2-2
Basic Oracle Architecture	2-4
The Large Pool	2-8
Data Buffer Cache, DBWn, and Data Files	2-11
Redo Log Buffer, LGWR, and Redo Log Files	2-14
Multiplexed Redo Log Files	2-17
CKPT Process	2-19
Fast-Start Checkpointing	2-21
Multiplexing Control Files	2-26
ARCn Process and Archived Log Files	2-28
Categories of Failures	2-31
Common Causes of Statement Failure	2-32

Resolutions for Statement Failures	2-33
Causes of User Process Failures	2-34
Resolving User Process Failures	2-35
Possible User Error Failures	2-36
Resolving User Errors	2-37
Instance Failure	2-38
Recovery from Instance Failure	2-39
Instance Recovery Process	2-41
Media Failure	2-43
Media Failure Resolution	2-44
Database Synchronization	2-45
Fast-Start Parallel Rollback	2-46
On-Demand Parallel Rollback	2-51
Quick Reference	2-52
Summary	2-53

Lesson 3: Oracle Backup and Recovery Configuration

Objectives	3-2
Redo Log	3-3
NOARCHIVELOG Mode	3-4
ARCHIVELOG Mode	3-6
Set Archivelog Destination	3-8
Duplexing Archived Log Files	3-9
Specifying Multiple Archive Locations	3-10
Multiple Archive Options	3-11
Specifying Minimum Number of Local Destinations	3-13
Controlling Archiving to a Destination	3-15
Enabling ARCHIVELOG Mode	3-16
Multiple Archive Processes	3-18
Enabling Archive Process	3-19
Enabling the Archive Process in an Open Instance	3-21
Enabling Archive Processes at Start of Instance	3-23
Stop or Start Additional Archive Processes	3-24

Disabling Archive Processing	3-25
Selectively Archiving Log Files	3-26
Obtaining Archive Log Information	3-28
Recovery Configuration	3-31
Monitoring Recovery Time	3-32
Summary	3-34
Quick Reference	3-35

Lesson 4: Physical Backups Without Oracle Recovery Manager

Objectives	4-2
Overview	4-4
Closed Database Backups	4-5
Obtaining Database File Information	4-8
Performing a Closed Database Backup	4-10
Opened Database Backup	4-12
Performing an Opened Database Backup	4-16
Data Dictionary Views	4-18
Backing Up a Control File	4-20
Read-Only Tablespace Backup	4-22
Read-Only Tablespaces	4-23
Logging and Nologging Options	4-24
Summary	4-25
Quick Reference	4-26

Lesson 5: Complete Recovery Without Recovery Manager

Objectives	5-2
Overview	5-3
Media Failure	5-4
Recovery: NOARCHIVELOG Mode	5-6
Restoring Data Files	5-7
Complete Recovery	5-8
Recovery by Using Archived Log Files	5-13
Locating Data Files	5-15
Recovery After Failure of Opened Database Backup	5-27

Clearing Corrupted Redo Logs	5-30
Loss of Inactive Redo Log File	5-32
Re-creating Redo Logs	5-34
Recovery Status Information	5-36
Summary	5-38
Quick Reference	5-39

Lesson 6: Incomplete Oracle Recovery with Archiving

Objectives	6-2
Overview	6-3
Reasons for Incomplete Recovery	6-4
Incomplete Recovery	6-5
RECOVER Command	6-7
Recovery Steps	6-8
Incomplete Recovery Guidelines	6-9
The Alert Log	6-11
Time-Based Recovery	6-12
Incomplete Recovery Using Until Time	6-14
Incomplete Recovery Using Until Cancel	6-16
Incomplete Recovery Using the Backup Control File	6-19
Loss of Current Online Redo Logs	6-22
Recovery Through Resetlogs	6-26
Summary	6-29
Quick Reference	6-30

Lesson 7: Oracle Export and Import Utilities

Objectives	7-2
Export and Import Utility Overview	7-3
Methods to Run the Export Utility	7-5
Export Modes	7-6
Command Line Export	7-7
Complete Export	7-9
Incremental Export	7-10
Cumulative Export	7-11

Incremental and Cumulative Export Benefits	7-12
Direct Path Export Concepts	7-13
Specifying Direct Path Export	7-14
Direct Path Export Features	7-15
Direct Path Export Restrictions	7-16
Export Utility Compatibility Issues	7-17
Using the Import Utility for Recovery	7-18
Import Modes	7-19
Command Line Import	7-20
Import Process Sequence	7-22
NLS Considerations	7-24
Tablespace Point-in-Time Recovery (TSPITR)	7-26
TSPITR Methods	7-28
TSPITR by Using Transportable Tablespaces	7-29
Summary	7-31
Quick Reference	7-32

Lesson 8: Additional Recovery Issues

Objectives	8-2
Fast-Start Recovery	8-3
Minimize Downtime	8-5
Parallel Recovery Operations	8-7
Starting a Database with a Missing Data File	8-10
Loss of Non-Essential Data File	8-12
Loss of Control Files	8-15
Recovering Control Files	8-16
Read-Only Tablespace Recovery	8-18
Read-Only Tablespace Recovery Issues	8-19
Summary	8-20
Quick Reference	8-21

Lesson 9: Oracle Utilities for Troubleshooting

Objectives	9-2
The Alert Log File	9-3
Oracle Trace Files	9-5
Corrupt Block Detection and Repair	9-7
Methods for Detecting Block Corruption	9-8

DB_BLOCK_CHECKSUM Parameter	9-10
DB_BLOCK_CHECKING Parameter	9-11
DBVERIFY Utility	9-12
The Command Line Interface	9-14
DBMS_REPAIR Package	9-16
Mark Corrupted Objects	9-19
Skipping Blocks	9-21
Indexes and Corrupt Tables	9-22
Limitations of DBMS_REPAIR	9-24
ANALYZE VALIDATE STRUCTURE Command	9-25
LogMiner Utility	9-27
LogMiner Analysis	9-29
Views	9-36
Limitations	9-37
Summary	9-38
Quick Reference	9-39

Lesson 10: Oracle Recovery Manager Overview

Objectives	10-2
Overview	10-3
Recovery Manager Features	10-4
Recovery Manager Components	10-6
Recovery Manager Packages	10-8
RMAN Setup Considerations	10-9
The Recovery Catalog	10-11
Control File Information	10-13
Connecting Without a Recovery Catalog	10-15
Recovery Manager Modes	10-17
RMAN Commands	10-19
Channel Allocation	10-21
REPORT Command	10-22
REPORT NEED BACKUP Command	10-23
LIST Command	10-24

Stored Scripts	10-26
RUN Command	10-30
Media Management	10-31
Summary	10-33
Quick Reference	10-34

Lesson 11: Oracle Recovery Catalog Creation and Maintenance

Objectives	11-2
Overview	11-3
Creating the Recovery Catalog	11-7
Connecting Using a Recovery Catalog	11-10
Catalog Maintenance	11-12
CHANGE Command	11-15
Deleting a Backup	11-17
Deleting Records of Previous Incarnation	11-19
Recovery Catalog Backup	11-20
Recovering the Recovery Catalog	11-21
Recovery Catalog Reporting	11-22
LIST Command	11-24
Stored Scripts	11-25
RESET DATABASE Command	11-29
Data Dictionary Views	11-31
Summary	11-33
Quick Reference	11-35

Lesson 12: Backups Using Recovery Manager

Objectives	12-2
Overview	12-3
Backup Concepts	12-4
Whole Database Backup	12-6
Recovery Manager Backup Types	12-8
Allocating a Channel	12-9
Tags	12-11
Image Copies	12-12

Image Copy Characteristics	12-13
COPY Command	12-14
Image Copy Process	12-15
Image Copy Parallelization	12-16
Image Copy of All Data Files	12-17
Monitoring the Copy Process	12-18
Operating System Copies	12-20
Backup Sets	12-21
BACKUP Command	12-22
Backup Set Characteristics	12-25
Multiplexed Backup Sets	12-26
Parallelization of Backup Sets	12-27
Backup Piece	12-29
Backup Piece Size	12-30
Data File Backup Process	12-31
Archived Log Backup Sets	12-32
Archived Log Backup	12-33
Full, Incremental, and Cumulative Backups	12-34
Incremental Backups	12-36
Multilevel Incremental Backup	12-38
Incremental Backups	12-39
Cumulative Incremental Backups	12-41
Backup Constraints	12-42
Backup Set Scenarios	12-43
Backups Using Stored Scripts	12-49
Miscellaneous Issues	12-51
Memory Usage by Recovery Manager	12-53
Troubleshooting	12-55
Data Dictionary Views	12-57
Summary	12-58
Quick Reference	12-59

Lesson 13: Restoration and Recovery Using Recovery Manager

- Objectives 13-2
- Restoration and Recovery Using Recovery Manager 13-3
- Restoring a Database in NOARCHIVELOG Mode 13-4
- Restoring Data Files to a Different Location 13-6
- Recovering a Tablespace 13-7
- Relocating a Tablespace 13-9
- Incomplete Recovery of a Database 13-12
- Incomplete Recovery of a Database: Example 13-13
- Restoring a Database to a Previous Incarnation 13-15
- Summary 13-17

Lesson 14: Oracle Standby Database

- Objectives 14-2
- Overview 14-3
- Standby Database Features 14-4
- Standby Database Guidelines 14-6
- Initialization Parameters 14-7
- Creating a Standby Database 14-8
- Managed Recovery Mode 14-10
- Maintaining the Standby Database 14-11
- Read Only Mode of Standby Database 14-12
- Activating the Standby Database 14-14
- Operating a Standby Database 14-16
- Structural Change of Primary Database 14-18
- Refreshing the Control File 14-20
- Nologging Operations at the Primary Database 14-21
- Summary 14-22
- Quick Reference 14-23

Lesson 15: Workshop

- Objectives 15-2
- Workshop Methodology 15-3
- Workshop Approach 15-5
- Business Requirements 15-6
- Resolving a Failure 15-7
- Summary 15-9

Appendix A: Practices

Practice Session Note	A-2
Practice 2-1	A-3
Practice 3-1	A-4
Practice 4-1	A-5
Practice 5-1	A-6
Practice 5-2	A-7
Practice 6-1	A-11
Practice 7-1	A-13
Practice 8-1	A-14
Practice 9-1	A-15
Practice 10-1	A-16
Practice 11-1	A-17
Practice 12-1	A-18
Practice 13-1	A-19

Appendix B: Practice Solutions

Practice Session Note	B-2
Practice 2-1 Solutions	B-3
Practice 3-1 Solutions	B-5
Practice 4-1 Solutions	B-9
Practice 5-1 Solutions	B-12
Practice 5-2 Solutions	B-14
Practice 6-1 Solutions	B-21
Practice 7-1 Solutions	B-27
Practice 8-1 Solutions	B-28
Practice 9-1 Solutions	B-31
Practice 10-1 Solutions	B-34
Practice 11-1 Solutions	B-35
Practice 12-1 Solutions	B-40
Practice 13-1 Solutions	B-42

Appendix C: Hints

Practice 2-1 Hints	C-2
Practice 3-1 Hints	C-4
Practice 4-1 Hints	C-6
Practice 5-1 Hints	C-7

Practice 5-2 Hints	C-8
Practice 6-1 Hints	C-10
Practice 7-1 Hints	C-12
Practice 8-1 Hints	C-13
Practice 9-1 Hints	C-15
Practice 11-1 Hints	C-16
Practice 12-1 Hints	C-18
Practice 13-1 Hints	C-19

Appendix D: Workshop Scenarios

Workshop Scenarios	D-2
Scenario 1: Loss of INACTIVE Online Redo Log Group	D-3
Scenario 2: Loss of CURRENT Online Redo Log Group	D-4
Scenario 3: Loss of Control Files	D-5
Scenario 4: Loss of Media	D-6
Scenario 5: Loss of File Containing Online Rollback Segment	D-7
Scenario 6: Loss of a Data File of System Tablespace	D-8
Scenario 7: Loss of a Non-System, Non-Rollback Segment Data File	D-9
Scenario 8: Recover from User Errors	D-10
Scenario 9: Failure During Hot Backup	D-11
Scenario 10: Configuring a Recovery Catalog	D-12
Scenario 11: Missing Data File with No Backup	D-14
Scenario 12: Loss of a Data File and Missing Archive Log File	D-15
Scenario 13: Loss of Non-Essential Data File When Database Is Down	D-16
Scenario 14: Recover a Lost Data File from Archive Logs	D-17
Scenario 15: Missing Mirrored Online Redo Log Files	D-18
Scenario 16: Loss of a Control File and Read-Only Tablespace	D-19

Appendix E: Worldwide Support Bulletins

Oracle Corporate Support Problem Repository	E-2
---	-----

Introduction

Objectives

Objectives

After completing this course, you should be able to do the following:

- **Understand Oracle's structures and processes for backup and recovery**
- **Learn the different Oracle backup and recovery methods**
- **Diagnose and troubleshoot database problems and failures**
- **Understand Oracle Recovery Manager**
- **Use workshop scenarios to recover Oracle databases from failure**

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE**

Preface

Profile

This course is designed to give the Oracle database administrator (DBA) a firm foundation in basic administrative tasks. The primary goal of this course is to give the DBA the necessary knowledge and skills to set up, maintain, and troubleshoot an Oracle database. This course has been designed for database administrators, technical support analysts, system administrators, application developers, MIS managers, and other Oracle users.

This preface covers the following sections:

- Before You Begin This Course
- Prerequisites
- How This Course Is Organized
- How This Book Is Organized
- Related Publications
- Typographic Conventions

Before You Begin This Course

The specific skills you as a participant must have in order to derive the maximum value from attending this course are:

- Familiarity with relational database concepts
- Thorough knowledge of SQL, SQL*Plus, and PL/SQL
- Basic knowledge of the operating system
- Working experience with Oracle

Prerequisites

- *SQL 1*
- *PL/SQL Fundamentals*

How This Course Is Organized

Enterprise DBA Part 1B: Backup and Recovery Workshop is an instructor-led course featuring lectures and hands-on exercises and a workshop. Online demonstrations, animation, and written practice sessions reinforce the concepts and skills introduced. The course also uses challenge-level practice labs including scenarios.

In addition, bulletins from Oracle Worldwide Support that address the most frequently asked questions are used to prepare participants to troubleshoot “real-world” issues.

This course contains clearly defined objectives designed to support preparation for the *Oracle Certified Professional* examination.

Related Publications

Oracle Publications

Title	Part Number
<i>Oracle8i Generic Documentation Set, Release 8.1.5</i>	A69148
<i>Oracle8i Enterprise Edition: Getting Started Release 8.1.5 for Windows NT</i>	A68694-02
<i>Oracle8i: Administrator's Reference Release 8.1.5 for Sun SPARC Solaris</i>	A67456-01
<i>Oracle8i: Backup and Recovery Guide</i>	A67773-01
<i>Oracle Press: Oracle8 - Backup and Recovery Handbook</i>	A54760
<i>Oracle Press: Oracle8 - The Complete Reference</i>	A54759

Web Sites

http://www.oracle.com
http://education.us.oracle.com
http://www.oramag.com

Oracle Education Student Union

In the Student Union (Oracle's online student community), you can continue to learn through online forums with Oracle instructors or online mini-lessons with streaming Real Video. You can also find valuable course information in the Resource Center.

Log in to the Oracle Education Student Union at <http://education.oracle.com> to view the following mini-lesson: Oracle8i-Availability and Recoverability

Related Additional Publications

- System release bulletins
- Installation and Configuration Guides
- International Oracle User's Group (IOUG) articles
- *Oracle Magazine*

Typographic Conventions

Typographic Conventions in Text

Convention	Element	Example
Bold italic	Glossary term (if there is a glossary)	The <i>algorithm</i> inserts the new key.
Caps and lowercase	Buttons, check boxes, triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger . . . Open the Master Schedule window.
Courier new, case sensitive (default is lowercase)	Code output, directory names, filenames, passwords, pathnames, URLs, user input, usernames	Code output: <code>debug.seti('I',300);</code> Directory: bin (DOS), \$FMHOME (UNIX) Filename: Locate the <code>init.ora</code> file. Password: Use <code>tiger</code> as your password. Pathname: Open <code>c:\my_docs\projects</code> URL: Go to <code>http://www.oracle.com</code> User input: Enter <code>300</code> Username: Log on as <code>scott</code>
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (<i>but</i> Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do <i>not</i> save changes to the database. For further information, see <i>Oracle7 Server SQL Language Reference Manual</i> . Enter <i>user_id</i> @us.oracle.com, where <i>user_id</i> is the name of the user.
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select "Include a reusable module component" and click Finish. This subject is covered in Unit II, Lesson 3, "Working with Objects."
Uppercase	SQL column names, commands, functions, schemas, table names	Use the SELECT command to view information stored in the LAST_NAME column of the EMP table.

Convention	Element	Example
Arrow	Menu paths	Select File—>Save.
Brackets	Key names	Press [Enter].
Commas	Key sequences	Press and release these keys one at a time: [Alt], [F], [D]
Plus signs	Key combinations	Press and hold these keys simultaneously: [Ctrl]+[Alt]+[Del]

Typographic Conventions in Code

Convention	Element	Example
Caps and lowercase	Oracle Forms triggers	When-Validate-Item
Lowercase	Column names, table names	SELECT last_name FROM s_emp;
	Passwords	DROP USER scott IDENTIFIED BY tiger;
	PL/SQL objects	OG_ACTIVATE_LAYER (OG_GET_LAYER ('prod_pie_layer'))
Lowercase italic	Syntax variables	CREATE ROLE <i>role</i>
Uppercase	SQL commands and functions	SELECT userid FROM emp;

Typographic Conventions in Navigation Paths

This course uses simplified navigation paths, such as the following example, to direct you through Oracle Applications.

(N) Invoice—>Entry—>Invoice Batches Summary (M) Query—>Find
(B) Approve

This simplified path translates to the following:

- 1 (N) From the Navigator window, select Invoice—>Entry—>Invoice Batches Summary.
- 2 (M) From the menu bar, select Query—>Find.
- 3 (B) Click the Approve button.

N = Navigator, **M** = Menu, **B** = Button

Backup and Recovery Considerations

Objectives

Objectives

After completing this lesson, you should be able to do the following:

- **Define requirements for a backup and recovery strategy**
- **Articulate the importance of management concurrence for the strategy**
- **Identify the components of a disaster recovery plan**
- **List Oracle Server features in the context of high availability**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Objectives

- **List the strengths of different database configurations for recoverability**
- **Discuss the importance of testing a backup and recovery plan**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Overview

Backup and Recovery Issues

- **Protect the database from numerous types of failures**
- **Increase Mean-Time-Between-Failures (MTBF)**
- **Decrease Mean-Time-To-Recover (MTTR)**
- **Minimize data loss**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Overview

One of a database administrator's (DBA) major responsibilities is to keep the database available for use. The DBA can take precautions to minimize failure of the system.

In spite of the precautions, it is naive to think that failures will never occur. The DBA must make the database operational as early as possible in case of a failure. The DBA can minimize the loss of data.

To protect the data from all the various types of failures that can occur, the DBA must back up the database regularly. Without a current backup, it is impossible for the DBA to get the database up and running if there is a file loss, without losing data.

Backups are critical for recovering from different types of failures. The task of validating the backups cannot be overemphasized. The presumption that the backup exists can prove very costly.

Defining a Backup and Recovery Strategy

Defining a Backup and Recovery Strategy

- **Business requirements**
- **Technical requirements**
- **Operational requirements**
- **Management concurrence**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Questions for the DBA

Whatever backup strategy you choose, it is important to obtain agreement from all appropriate levels of management. For example, if your company wants to avoid taking physical image copies of the files to minimize usage of disk space, management must be aware of the ramifications this decision will pose.

Here are some questions to consider when selecting a backup strategy:

- Does management understand the tradeoffs involved in their expectations of system availability?
- Is management willing to dedicate the resources needed to ensure a successful backup and recovery strategy?
- Does management understand the importance of making backups and prepare recovery procedures?

Performing a thorough analysis of the business, operational, and technical requirements provides management with the information they need to support an effective backup and recovery strategy.

Business Requirements

Business Requirements

- **Mean-Time-To-Recover**
- **Mean-Time-Between-Failure**
- **Evolutionary process**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Business Impact

You should realize the impact that downtime will have on the business. Management must quantify the cost of downtime and the loss of data and compare this with the cost of reducing downtime and minimizing data loss.

MTTR Database availability is a key issue for a DBA. In the event of a failure the DBA should strive to reduce the Mean-Time-To-Recover (MTTR). This strategy ensures that the database is unavailable for the shortest possible amount of time. Anticipating the types of failures that can occur and using effective recovery strategies, the DBA can ultimately reduce the MTTR.

MTBF Protecting the database against various types of failures is also a key DBA task. To do this, a DBA must increase the Mean-Time-Between-Failures (MTBF). The DBA must understand the backup and recovery structures within an Oracle database environment and configure the database so that failures will not occur often.

Evolutionary Process A backup and recovery strategy evolves as business, operational, and technical requirements change. It is important that both the DBA and management review the validity of a backup and recovery strategy on a regular basis.

Operational Requirements

Operational Requirements

- 24-hour operations
- User and operator appreciation
- Testing and validating backups

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

24-Hour Operations

Backups and recoveries are always affected by the type of business operations, particularly in a situation where a database must be available 24 hours a day, 7 days a week for continuous operation. Proper database configuration is necessary to support these operational requirements because they directly affect the technical aspects of the database environment.

Testing Backups

DBAs can ensure that they have a strategy that enable them to decrease the MTTR and increase the MTBF having a plan in place to test the validity of backups regularly. A recovery is only as good as the backups that are available. Here are some questions to consider when selecting a backup strategy:

- Can I depend on system administrators, vendors, backup DBAs, and so forth when I need help?
- Can I test my backup and recovery strategies at frequently scheduled intervals?
- Are backup copies stored off-site?
- Is a plan well documented and maintained?

Database Volatility

Other issues that impact operational requirements include the volatility of the data and structure of the database. Here are some questions to consider when selecting a backup strategy:

- Are tables frequently updated?
- Is data highly volatile? If so, you will need backups more frequently than a business where data is relatively static.
- Does the structure of the database change often?
- How often do you add data files?

Technical Requirements

Technical Requirements

- **Resources:** Hardware, software, manpower, and time
- **Physical image copies of the operating system files**
- **Logical copies of the objects in the database**
- **Database configurations**
- **Transaction volume affects desired frequency of backups**

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Physical Image Copies

Certain technical requirements are dictated by the types of backups that are required. For example, if physical image copies of data files are required, this may significantly impact available storage space.

Logical Copies

Performing logical copies of objects in the database may not have as significant storage requirements as physical image copies; however, system resources may be affected since logical copies are performed while the database is being accessed by users.

Database Configuration

Different database configurations affect how backups are performed and the availability of the database. Depending on the database configuration, system resources such as disk space required to support a backup and recovery strategy may be limited.

Transaction Volumes

Transaction volumes also affect system resources. If 24-hour operations require regular backups, the load on system resources is increased.

Technical Requirements

Here are some questions to consider when selecting a backup strategy:

- How much data do you have?
- Do you have the machine power and capacity to support backups?
- Is the data easily recreated?
- Can you reload the data into the database from a flat file?
- Does the database configuration support resiliency to different types of failures?

Disaster Recovery Issues

Disaster Recovery Issues

- **How will your business be affected in the event of a major disaster?**
 - Earthquake, flood, fire, or complete loss of machine
 - Malfunction of storage hardware or software
 - Loss of key personnel, such as the database administrator
- **Do you have a plan for testing your strategy periodically?**
- **Do you perform the strategy tests?**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Natural Disaster

Perhaps your data is so important that you must ensure resiliency even in the event of a complete system failure. Natural disasters and other issues can affect the availability of your data and must be considered when creating a disaster recovery plan. Here are some questions to consider when selecting a backup strategy:

- What will happen to your business in the event of a serious disaster such as:
 - Flood, fire, earthquake, or hurricane
 - Malfunction of storage hardware or software
- If your database server fails, will your business be able to operate during the hours, days, or even weeks it might take to get a new hardware system?
- Do you store backups off-site?

Solutions

- Off-site backups
- Standby Database feature that enables a DBA to fall back on another database that is configured as a standby in case the primary database fails.
- Geomirroring
- Messaging
- TP monitors

Loss of Key Personnel

In terms of key personnel, consider the following questions:

- How will a loss of personnel affect your business?
- If your DBA leaves the company or is unable to work, will your database system continue to run?
- Who will handle a recovery situation if the DBA is unavailable?

Oracle Availability and Features

Oracle Features

Oracle features for maintaining high availability of database include:

- Oracle Parallel Server
- Oracle FailSafe

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Oracle Parallel Server

Oracle Parallel Server is an optional feature that enables multiple database instances to use one single database on a cluster. So, when one node fails, another node can take over the tasks of the first node. The implementation of Oracle Parallel Server is discussed in detail in a separate course, *Implementing Parallel Server*.

Oracle FailSafe

The Oracle FailSafe feature is available on WindowsNT platforms only. In this environment, two nodes share a disk system on which a database is located. At any one point, only one instance is operational. When the instance that is operational fails, the other node instantiates (starts the instance) automatically.

Summary

Summary

In this lesson, you should have learned how to:

- **Develop a strategy dictated by business, operational, and technical requirements**
- **Consider a test plan for a backup and recovery strategy**
- **Constitute an effective strategy**
- **Take advantage of new Oracle8i features for availability**

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

2

Oracle Recovery Structures and Processes

Objectives

Objectives

After completing this lesson, you should be able to do the following:

- List Oracle processes, memory, and file structures relating to recovery
- Identify the importance of checkpoints, redo logs, and archives
- Multiplex control files and redo logs

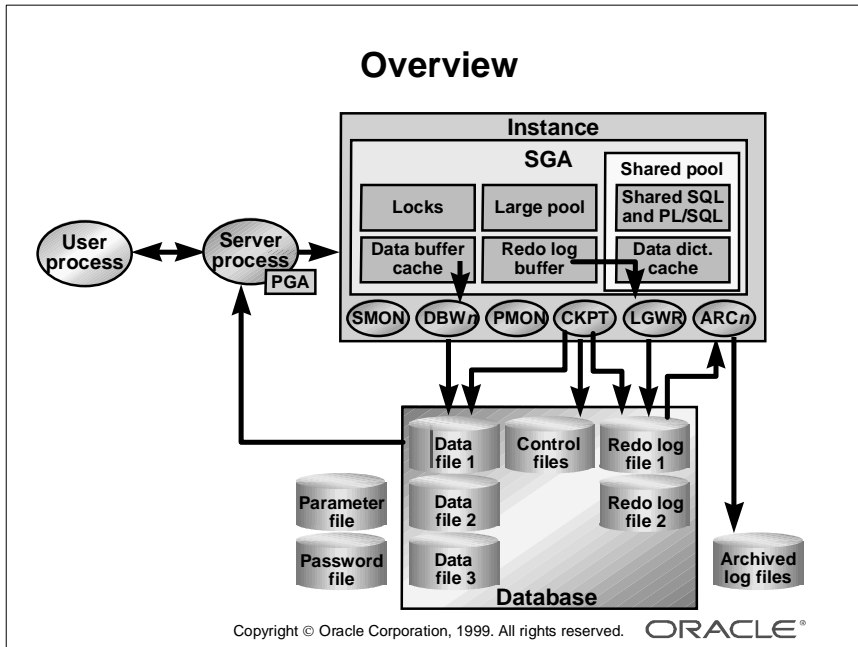
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Objectives

- **List the types of failure**
- **Describe the structures for instance and media recovery**
- **Describe the deferred transaction recovery concept**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Basic Oracle Architecture



Overview

The RDBMS uses many memory components, background processes, and file structures for its backup and recovery mechanism. This lesson reviews the concepts presented in the *Oracle8i: Database Administration* course, with an emphasis on backup and recovery requirements.

Oracle Instance

An Oracle instance consists of memory areas (mainly System Global Area [SGA]) and background processes, namely PMON, SMON, DBW_n, LGWR, and CKPT. An instance is created during the nomount stage of database startup after the parameter file has been read. If any of these processes terminate, the instance shuts down. While the five processes referred to above are essential for the startup of an instance, to operationalize an instance effectively, more processes such as the user processes and server processes are required.

Memory Structures

Type	Description
Data buffer cache	Memory area used to store blocks read from data files. Data is read into the blocks by server process and written out by DBWn asynchronously.
Log buffer	Memory containing before and after image copies of changed data to be written to the redo logs.
Large pool	An optional memory area used for I/O by RMAN backup and restore is discussed in more detail later in this lesson.
Shared pool	Stores parsed versions of SQL statements, PL/SQL procedures, and data dictionary information.

Background Processes

Type	Description
Database writer (DBWn)	Writes dirty buffers from the data buffer cache to the data files. This activity is asynchronous.
Log writer (LGWR)	Writes data from the redo log buffer to the redo log files.
System monitor (SMON)	Performs automatic instance recovery. Recovers space in temporary segments when they are no longer in use. Merges contiguous areas of free space depending on parameters set.
Process monitor (PMON)	Cleans up the connection/server process dedicated to an abnormally terminated user process. Performs rollback and releases the resources held by the failed process.
Checkpoint (CKPT)	Synchronizes the headers of the data files and control files with the current redo log and checkpoint numbers.
Archiver (ARCn) (optional)	A process that automatically copies redo logs that have been marked for archiving.

The User Process

The user process is created when a user starts a tool such as SQL*Plus, Forms, Reports, Enterprise Manager, and so on. This process might be on the client or server, and provides an interface for the user to enter commands that interact with the database.

The Server Process

The server process accepts commands from the user process and performs steps to complete user requests. If the database is not in a multithreaded configuration, a server process is created on the machine containing the instance when a valid connection is established.

Oracle Database

An Oracle database consists of the physical files.

File Type	Description	Type
Data files	Physical storage of data. At least one file is required per database. This file stores the system tablespace.	Binary
Redo logs	Contain before and after image copies of changed data, for recovery purposes. At least two groups are required.	Binary
Control files	Record the physical structure and status of the database.	Binary
Parameter file	Store parameters required for instance startup.	Text
Password file (optional)	Store information on users who can start, stop, and recover the database.	Binary
Archive logs (optional)	Physical copies of the online redo log files. Created when the database is set in ARCHIVELOG mode. Used in recovery.	Binary

Dynamic Views

The Oracle server provides a number of standard data dictionary views to obtain information on the database and instance. These views include:

- V\$SGA: Queries the size of the instance for the shared pool, log buffer, data buffer cache, and fixed memory sizes (operating system dependent).
- V\$INSTANCE: Queries the status of the instance, such as the instance mode, instance name, startup time, and host name.
- V\$PROCESS: Queries the background and server processes created for the instance.
- V\$BGPROCESS: Queries the background processes created for the instance.

Dynamic Views (continued)

- **V\$DATABASE:** Lists status and recovery information about the database. It includes information on the database name, the unique database identifier, the creation date, the control file creation date and time, the last database checkpoint, and other information.
- **V\$DATAFILE:** Lists the location and names of the data files that are contained in the database. It includes information relating to the file number and name, creation date, status (online/off-line), enabled (read-only, read-write), last data file checkpoint, size, and other information.

The Large Pool

Large Pool

- Can be configured as a separate memory area in the SGA, used for memory with:
 - I/O slaves: **DBWR_IO_SLAVES**
 - Oracle backup and restore: **BACKUP_TAPE_IO_SLAVES**
 - Session memory for the multi-threaded servers
- Is sized by the **LARGE_POOL_SIZE** parameter

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

The Large Pool

The large pool is used to allocate sequential I/O buffers from shared memory. For I/O slaves and Oracle backup and restore, the RDBMS allocates buffers that are a few hundred kilobytes in size.

Recovery Manager (RMAN) uses the large pool for backup and restore when you set the **DBWR_IO_SLAVES** or **BACKUP_TAPE_IO_SLAVES** parameters.

Sizing the Large Pool

If the **LARGE_POOL_SIZE** initialization parameter is not set, then the Oracle server attempts to allocate shared memory buffers from the shared pool in the SGA. If **LARGE_POOL_SIZE** is set but is not large enough, the allocation fails and the Oracle server component requesting the buffers does the following:

The RMAN command writes a message to the alert file and does not use I/O slaves for that operation.

Large Pool Parameters

- **LARGE_POOL_SIZE:** If this parameter is not set, then there is no large pool. The specified size of memory is allocated from the SGA.
 - Description: Size of the large pool, in bytes (can specify values in K or M)
 - Minimum: 300 K
 - Maximum: At least 2 GB (the maximum is operating system specific)
 - To determine how the large pool is being used, query V\$SGASTAT:

```
SQL> SELECT *
      2 FROM v$sgastat
      3 WHERE pool = 'large pool';
```

POOL	NAME	BYTES
large pool	free memory	4194304*

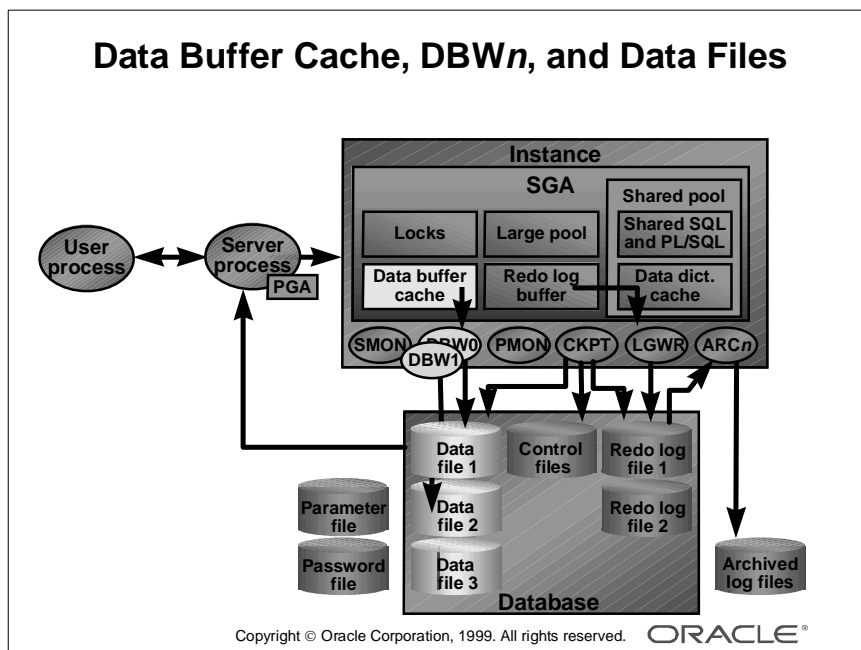
- **DBWR_IO_SLAVES:** This parameter specifies the number of I/O slaves used by the DBW_n process. The DBW_n process and its slaves always write to disk. By default, the value is 0 and I/O slaves are not used.
 - If DBWR_IO_SLAVES is set to a nonzero value, the numbers of I/O slaves used by the ARC_n process, LGWR process, and Recovery Manager are set to 4.
 - Typically, I/O slaves are used to simulate asynchronous I/O on platforms that do not support asynchronous I/O or implement it inefficiently. However, I/O slaves can be used even when asynchronous I/O is being used. In that case, the I/O slaves will use asynchronous I/O.
- **BACKUP_TAPE_IO_SLAVES:** It specifies whether I/O slaves are used by the Recovery Manager to backup, copy, or restore data to tape.
 - When BACKUP_TAPE_IO_SLAVES is set to TRUE, an I/O slave process is used to write to or read from a tape device.
 - If this parameter is set to FALSE (the default), then I/O slaves are not used for backups; instead, the shadow process engaged in the backup will access the tape device.

Large Pool Parameters (continued)

Note: Because a tape device can only be accessed by one process at any given time, this parameter is a Boolean, which allows or does not allow the deployment of an I/O slave process to access a tape device.

- In order to perform duplexed backups, this parameter needs to be enabled, otherwise an error will be signalled. Recovery Manager will configure as many slaves as needed for the number of backup copies requested when this parameter is enabled.

Data Buffer Cache, DBWn, and Data Files



Function of the Data Buffer Cache

- The *data buffer cache* is an area in the SGA that is used to store the most recently used data blocks.
- The server process reads tables, indexes, and rollback segments from the data files into the buffer cache where it makes changes to data blocks when required.
- The Oracle server uses a least recently used (LRU) algorithm to determine which buffers can be overwritten to accommodate new blocks in the buffer cache.

Function of the DBWn Background Process

- The *database writer process* (DBWn) writes the dirty buffers from the database buffer cache to the data files. It ensures that sufficient number of *free buffers*—buffers that can be overwritten when server processes need to read in blocks from the data files—are available in the database buffer cache.

Function of the DBW n Background Process (continued)

- The database writer regularly synchronizes the database buffer cache and the data files: this is the checkpoint event triggered in various situations.
- Although one DBW n is adequate for most systems, you can configure additional processes (DBW1 through DBW9) to improve write performance if your system modifies data heavily. These additional DBW n processes are not useful on uniprocessor systems.

Data Files

Data files store both system and user data on a disk. This data may be committed or uncommitted.

Data Files Containing Only Committed Data

This is normal for a closed database, except when failure has occurred or the “shutdown abort” option has been used. If the instance is shutdown and is using either normal or immediate or transactional, the data files contain only committed data. This is because all uncommitted data is rolled back, and a checkpoint is issued to force all committed data to a disk.

Data Files Containing Uncommitted Data

While an instance is running, data files can contain uncommitted data. This happens when data is changed but not committed (the data is now in the cache), and more space is needed in the cache (uncommitted data is forced off to a disk). Only when all users eventually commit will the data files contain only committed data. In the event of failure, during subsequent recovery, the redo logs and rollback segments are used to synchronize the data files.

Configuring Tablespaces

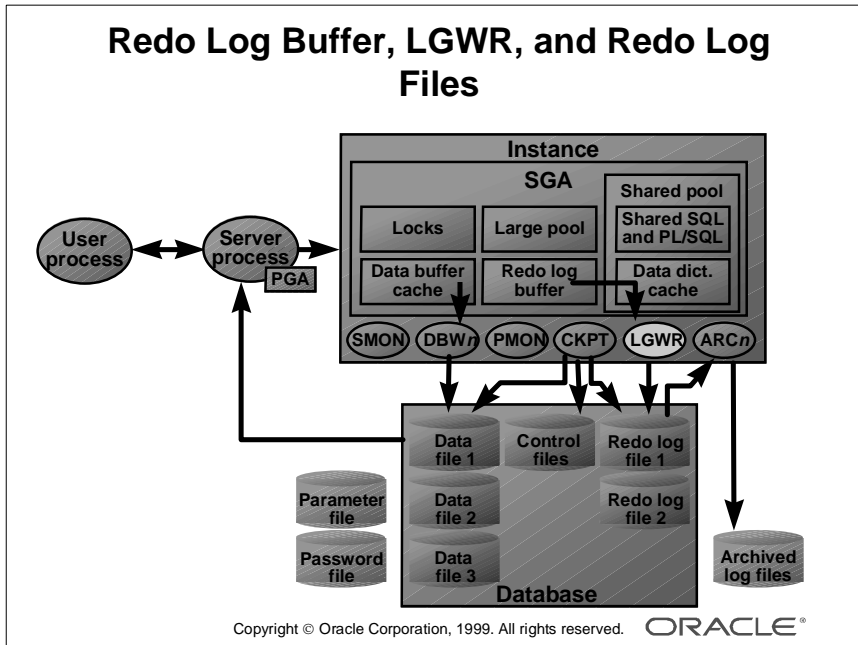
Tablespaces contain one or more data files. It is important that tablespaces are created carefully to provide a flexible and manageable backup and recovery strategy. Here are some examples of tablespaces:

- **SYSTEM:** Backup and Recovery are more flexible if system and user data is contained in different tablespaces.
- **TEMPORARY:** If the tablespace containing temporary segments (used in sort, and so on) is lost, it can be re-created, rather than recovered.

Configuring Tablespaces (continued)

- **ROLLBACK SEGMENTS:** Tablespaces containing online rollback segments are difficult to back up and recover with the database online. Such tablespaces should be dedicated to contain only rollback segments, such as the tablespace **SYSTEM**, which should contain only system and no application segments.
- **READ ONLY DATA:** Backup time can be reduced because a tablespace needs to be backed up only when the tablespace is made read-only.
- **HIGHLY VOLATILE DATA:** This tablespace can be backed up more frequently, also reducing recovery time.
- **INDEX DATA:** Tablespaces to store index segments should be created. These tablespaces can often be re-created instead of recovered.

Redo Log Buffer, LGWR, and Redo Log Files



Function of the Redo Log Buffer

- The redo log buffer is a circular buffer that holds information about changes made to the database. This information is stored in redo entries.
- Redo entries contain the information necessary to reconstruct, or redo, changes made to the database by INSERT, UPDATE, DELETE, CREATE, ALTER, or DROP operations. Redo entries are used for database recovery, if necessary.
- Redo entries are copied by Oracle server processes from the user's memory space to the redo log buffer.

Function of the LGWR Background Process

The *log writer (LGWR)* writes redo entries to the redo log files that have been copied into the redo log buffer since the last time it wrote to the redo log files:

- When the redo log buffer is one-third full
- When a timeout occurs (every three seconds)
- Before DBWn writes modified blocks in the database buffer cache to the data files
- When a transaction commits
- As long as the database, if obliged to archive the redo log files, did archive the redo log to be overwritten again

Redo Log Files

Redo log files store all changes made to the database. If the database is to be recovered to a point in time when it was operational, redo logs are used to ensure that all committed transactions are committed to disk, and all uncommitted transactions are rolled back. The important points relating to redo logs are as follows:

- LGWR writes to redo log files in a circular fashion. This behavior results in all members of a logfile group being overwritten.
- While it is mandatory to have at least two log groups to support the cyclic nature, in most cases, you would need more than just two redo log groups.

You can create additional log file groups using the following SQL command:

```
ALTER DATABASE [database]
    ADD LOGFILE [GROUP integer] filespec
    [,          [GROUP integer] filespec]...
```

To drop an entire online redo log group, use the following SQL command:

```
ALTER DATABASE [database]
    DROP LOGFILE
        {GROUP integer|('filename'[, 'filename']...)}
    [, {GROUP integer|('filename'[, 'filename']...)}]...
```

- To avoid a single-point media failure, it is recommended that you multiplex redo logs.

Redo Log Switches

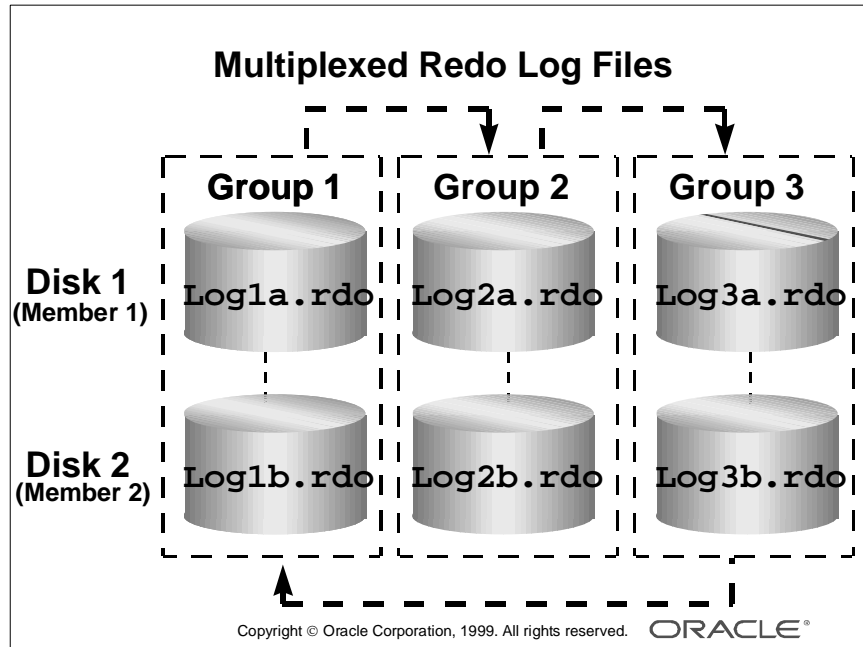
At a log switch, the current redo log group is assigned a log sequence number that identifies the information stored in that redo log group and is also used for synchronization.

- A log switch occurs when LGWR stops writing to one redo log group and begins writing to another.
- A log switch occurs when LGWR has filled one log file group.
- A DBA can force a log switch by using the ALTER SYSTEM SWITCH LOGFILE command.
- A checkpoint automatically occurs at a log switch.
- Processing can continue as long as at least one member of a group is available. If a member is damaged or unavailable, messages are written to the LGWR trace file and to the alert file.

Dynamic Views

- V\$LOG: Lists the number of members in each group. It contains:
 - The group number
 - The current log sequence number
 - The size of the group
 - The number of mirrors
 - Status (CURRENT or INACTIVE)
 - The checkpoint change numbers
- V\$LOGFILE: Lists the names, status (STALE or INVALID), and group of each log file member.
- V\$LOG_HISTORY: Contains information on log history from the control file.

Multiplexed Redo Log Files



Guidelines for Multiplexing

The redo log file configuration requires at least two redo log members per group, with each member on a different disk to guard against failure.

The locations of the online redo log files can be changed by renaming the online redo log files. Before renaming the online redo log files, make sure that the new online redo log file exists. The Oracle server changes only the pointers in the control files, but does not physically rename or create any operating system files.

How to Relocate a Redo Log File

- 1 If the log file is current, perform a log switch by using:

```
ALTER SYSTEM SWITCH LOG FILE;
```
- 2 Copy the redo log file from the previous location to the new location by using the operating system copy utility (cp in UNIX or COPY in NT)

Guidelines for Multiplexing (continued)

How to Relocate a Redo Log File (continued)

- 3** Use the ALTER DATABASE RENAME FILE command to make the change in control files:

```
ALTER DATABASE [database]
    RENAME FILE 'filename'[, 'filename']...
    TO 'filename'[, 'filename']...
```

- All members of a group contain identical information and are of the same size.
- Group members are updated simultaneously.
- Each group should contain the same number of members of the same size.

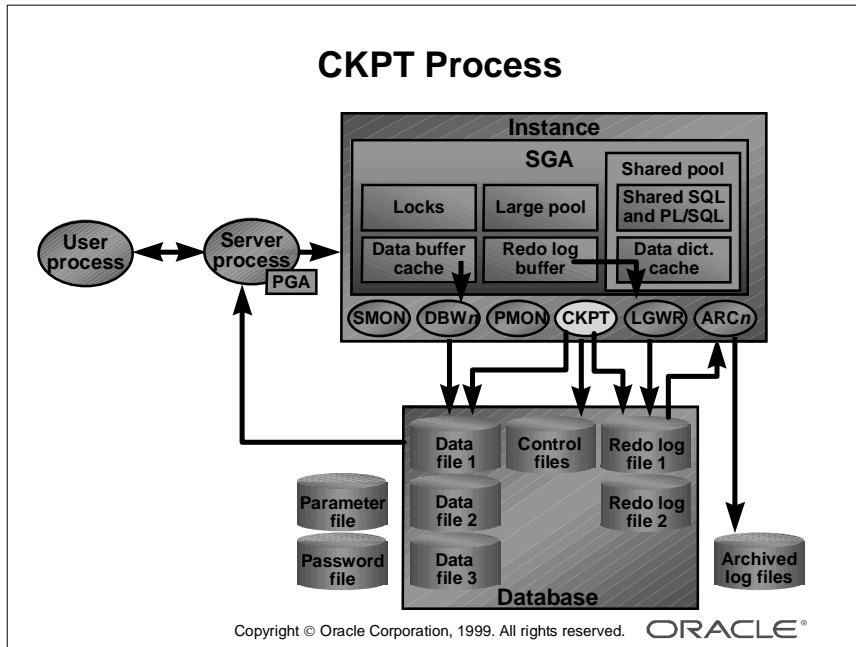
How to Add a Member to a Group You can add new members to existing redo log file groups by using the following SQL command:

```
ALTER DATABASE [database]
    ADD LOGFILE MEMBER
    [      'filename' [REUSE]
      [, 'filename' [REUSE]]...
    TO {GROUP integer
        | ('filename'[, 'filename']...)}
    }
```

How to Drop a Member from a Group You may want to drop an online redo log member if it becomes INVALID. Use the following command:

```
ALTER DATABASE [database]
    DROP LOGFILE MEMBER 'filename'[, 'filename']...
```


CKPT Process



Database Checkpoints

Database checkpoints ensure that all modified database buffers are written to the database files. The database header files are then marked current, and the checkpoint sequence number is recorded in the control file. Checkpoints synchronize the buffer cache by writing all buffers to disk whose corresponding redo entries were part of the log file being checkpointed.

CKPT Features

- The checkpoint process is always enabled.
- The CKPT process updates file headers at checkpoint completion.
- More frequent checkpoints reduce the time necessary for recovering from instance failure at the possible expense of performance.

When Does Checkpointing Occur?

- At every log switch (cannot be suppressed)
- When fast-start checkpointing is set to force DBWn to write buffers in advance in order to shorten the instance recovery
- When more than a fixed number of redo blocks defined by the LOG_CHECKPOINT_INTERVAL will need to be read during instance recovery
- When the elapsed time since writing the redo block at the current checkpoint position exceeds the number of seconds specified by the LOG_CHECKPOINT_TIMEOUT init.ora parameter.
- At instance shutdown, unless the instance is aborted
- When forced by a database administrator (ALTER SYSTEM CHECKPOINT command)
- When a tablespace is taken offline or an online backup is started

Note: Read-only data files are an exception: Their checkpoint numbers are frozen and do not correspond with the number in the control file.

Synchronization

- At each checkpoint, the checkpoint number is updated in every database file header and in the control file.
- The checkpoint number acts as a synchronization marker for redo, control, and data files. If they have the same checkpoint number, the database is considered to be in a consistent state.
- The control file confirms that all files are at the same checkpoint number during database startup. Any inconsistency between the checkpoint numbers in the various file headers results in a failure, and the database cannot be opened. Recovery is required.

Instance Recovery

Checkpoints expedite instance recovery because at every checkpoint all changed data is written to a disk. Once data resides in data files, redo log entries before the last checkpoint need not be applied again during the “roll forward” phase of instance recovery.

Fast-Start Checkpointing

Fast-Start Checkpointing Architecture

New dynamic parameter to limit data file I/O during recovery:

```
fast_start_io_target = 10000
```

- Useful in establishing service-level agreements with users
- Used in conjunction with other parameters to determine target for checkpointing

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Fast-Start Checkpointing

Prior to Oracle8i, it was difficult to control the amount of time an instance takes to perform instance recovery because it was dependent on transaction load at the time of failure.

Fast-Start Checkpointing can influence recovery performance for situations where there are stringent limitations on the duration of crash or instance recovery.

The time required for crash or instance recovery is roughly proportional to the number of data blocks that need to be read or written during the roll forward phase. You can specify a limit, or bound, on the number of data blocks that will need to be processed during roll forward. The Oracle server automatically adjusts the checkpoint write rate to meet the specified roll forward bound while issuing the minimum number of writes.

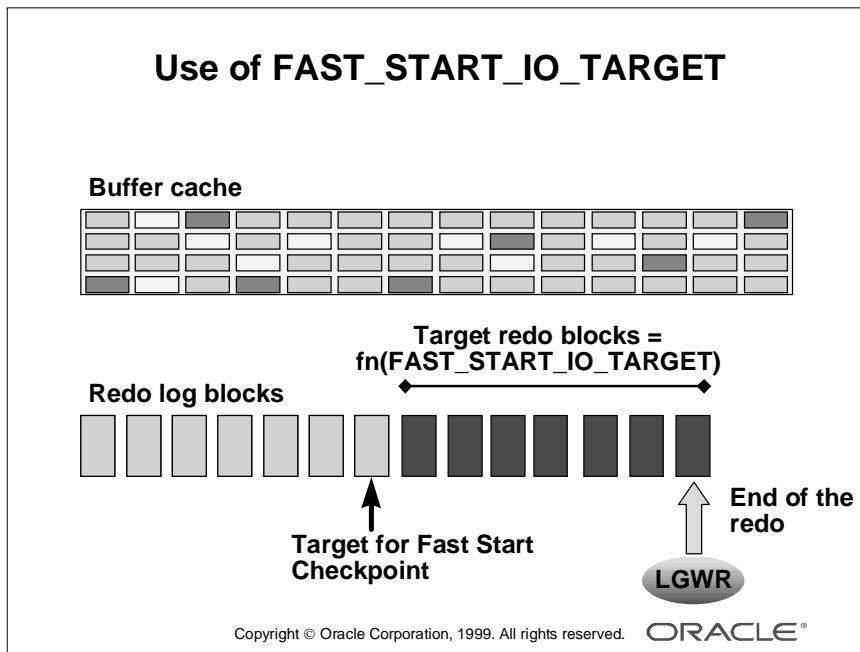
Since the time to recover from an instance failure is primarily dependent on data file I/O, and the average I/O time can be estimated from instance statistics, this parameter allows a DBA to establish service level agreements with users.

Fast-Start Checkpointing improves the performance of crash and instance recovery, but not media recovery.

Initialization Parameter

You can set the dynamic initialization parameter `FAST_START_IO_TARGET` to limit the number of blocks that need to be read for crash or instance recovery.

- Smaller values of this parameter impose higher overhead during normal processing because more buffers have to be written. On the other hand, the smaller the value of this parameter, the better the recovery performance, since fewer blocks need to be recovered.
- The dynamic initialization parameters `LOG_CHECKPOINT_INTERVAL` and `LOG_CHECKPOINT_TIMEOUT` also influence Fast-Start Checkpointing.



Fast-Start Checkpoint

Fast-Start Checkpoints cause database writer (DBW_n) to write blocks from the buffer cache so that the earliest buffer to be dirtied gets written first.

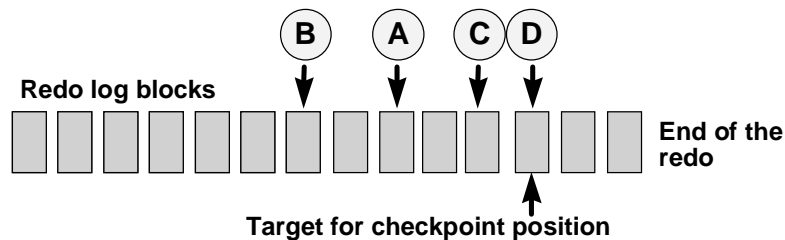
Fast-start checkpointing causes DBW_n to write data blocks continually so the checkpoint position in the redo log can advance and satisfy the target set by the `FAST_START_IO_TARGET` initialization parameter.

Impact of FAST_START_IO_TARGET on Checkpoints

`FAST_START_IO_TARGET` defines the number of I/O operations (data blocks) the Oracle server will need to process (read and write) during recovery. The Oracle server automatically examines the blocks in the redo log and dynamically calculates the target redo blocks given a value of `FAST_START_IO_TARGET`. The Oracle server then continually writes dirty buffers to advance the checkpoint position such that the number of redo records that would need to be processed in the event of recovery (those between the checkpoint position and the end of the redo log) does not exceed the calculated target redo blocks.

Other Factors Affecting Checkpointing

- A Target based on FAST_START_IO_TARGET
- B 90% of size of smallest redo log
- C End of the log LOG_CHECKPOINT_TIMEOUT seconds ago
- D LOG_CHECKPOINT_INTERVAL blocks from the end



Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Other Factors Affecting Checkpointing

In addition to FAST_START_IO_TARGET, a number of other factors also influence the target redo block for the checkpoint position. These include:

- 90% of the size of the smallest log file: A redo log file cannot be reused until the checkpoint has advanced beyond the end of the log file. To minimize the chance that update activity in the instance waits for a checkpoint to complete, Oracle8i makes sure that the target for checkpoint position does not lag the end of the redo log by more than 90% of the size of the smallest redo log.

The smaller the size of the smallest log, the more aggressively the Oracle server writes dirty buffers to disk to ensure the position of the checkpoint has advanced to the current log before that log completely fills.

Note: Although you specify the number and sizes of online redo log files at database creation, you can alter the characteristics of your redo log files after startup. Use the DROP LOGFILE clause of the ALTER DATABASE command to drop the redo log first, then use the ADD LOGFILE clause to re-create the redo log file with a new size.

The size of the redo log appears in the LOG_FILE_SIZE_REDO_BKLS column of the V\$INSTANCE_RECOVERY dynamic performance view. This value shows how the size of the smallest online redo log is affecting checkpointing. By increasing or decreasing the size of your online redo logs, you indirectly influence the frequency of checkpoint writes.

Other Factors Affecting Checkpointing (continued)

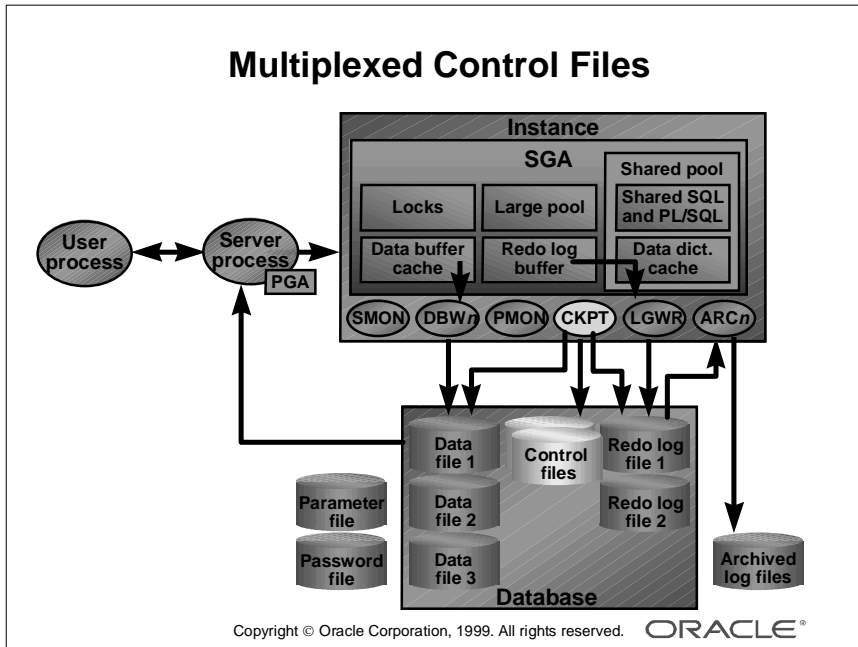
- **LOG_CHECKPOINT_INTERVAL:** In Oracle8i, when LOG_CHECKPOINT_INTERVAL is set, the target for checkpoint position cannot lag the end of the log by more than the number of redo log blocks specified by this parameter. This ensures that no more than a fixed number of redo blocks will need to be read during instance recovery.
- **LOG_CHECKPOINT_TIMEOUT:** When specified, this parameter sets the target for checkpoint position to a location in the log file where the end of the log was this many seconds ago. This ensures that no more than the specified number of seconds worth of redo log blocks need to be read during instance recovery.

Database writer considers all of these factors, as applicable, and uses the most aggressive point as the target for checkpoint position. By choosing the point closest to the end of the redo log based on these factors, all the criteria defined are satisfied.

Note: In release 8.0, LOG_CHECKPOINT_INTERVAL was used to specify the number of redo blocks after which to initiate a checkpoint, and LOG_CHECKPOINT_TIMEOUT was used to specify the time in seconds to initiate a checkpoint.

Another parameter DB_BLOCK_MAX_DIRTY_TARGET indirectly specifies a rough limit on the number of blocks that must be read during crash and instance recovery since it specifies the number of buffers that can be dirty (modified and different from what is on disk) in the buffer cache: DBWn will write out buffers to attempt to limit the number of dirty buffers in the cache below the specified value. But FAST_START_IO_TARGET gives better control over recovery time and is recommended.

Multiplexing Control Files



Control File Function

The control file is a small binary file that describes the structure of the database. It must be available for writing by the Oracle server whenever the database is open and its default name is operating system dependent. Without this file, the database cannot be mounted and recovery is difficult.

Control File Properties

- All necessary database files and log files are identified.
- The name of the database is stored.
- The control file is required to mount, open, and maintain the database.
- Synchronization information (checkpoint and log sequence information) needed for recovery is stored.
- The recommended configuration is a minimum of two control files on different disks.
- Time stamp of database creation is stored.
- Extra backup information is stored when using Recovery Manager.

Dynamic View

To obtain the location and names of the control files, use either the dynamic performance view V\$PARAMETER or the dynamic performance view V\$CONTROLFILE.

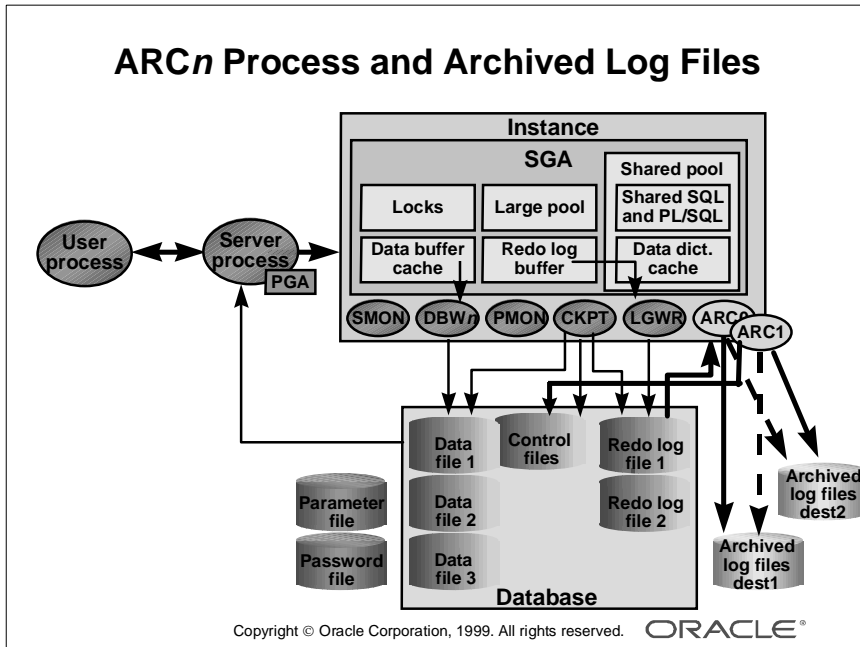
```
SQL> SELECT name
2  > FROM v$controlfile;
NAME
-----
/DISK1/control01.con
/DISK2/control02.con
2 rows selected.
```

How to Multiplex the Control File

To add a new control file or change the number or location of the control file, follow these steps:

- 1** Shut down the database.
- 2** Make a copy of the existing control file to a different device by using operating system commands.
- 3** Edit or add the CONTROL_FILES parameter and specify names for all the control files.
- 4** Start the database.

ARC_n Process and Archived Log Files



Function of the Archive Background Process

The ARC_n process is an optional process. When enabled, it archives the redo log files to the designated storage areas. This process has a great significance in backup, restoration, and recovery of a database set to Archivelog mode, where databases are operational 24 hours a day and 7 days a week.

The ARC_n process initiates when a log switch occurs and copies one member of the last (unarchived) redo log group to at least one of the destinations specified by some `init.ora` parameters.

Archived Log Files

When the database is set to ARCHIVELOG mode, the LGWR process waits for the online redo log files to be archived (either manually or through the ARCn process) before they can be reused.

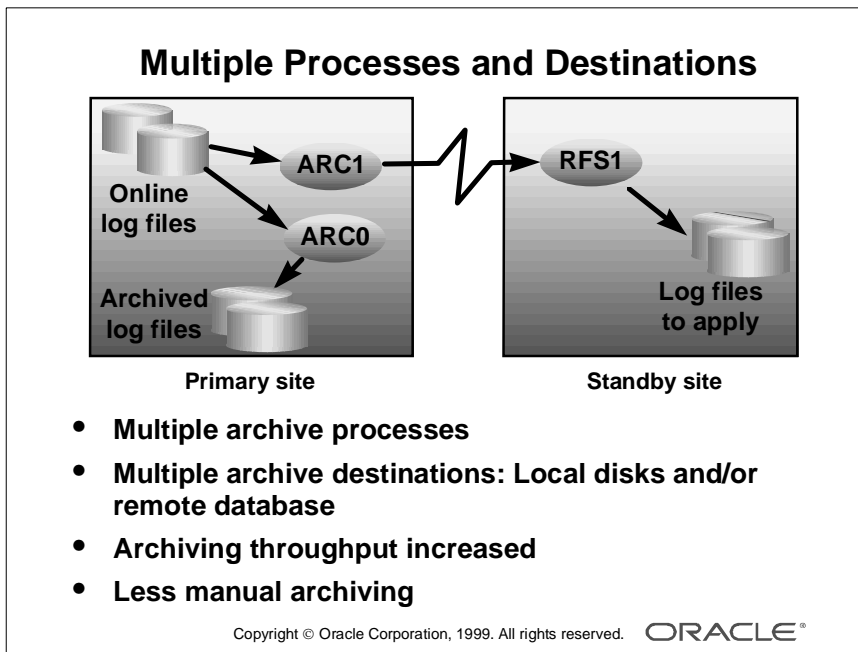
If online redo log file is corrupt, another member from the same group is used.

Archived logs are beneficial to the backup and recovery process because:

- A database backup, combined with archived redo log files, guarantees that all committed data can be recovered to the point of failure.
- Valid database backups can be taken while the database is online.

Archiving Considerations

The choice of whether to enable archiving depends on the availability and reliability requirements of each database. Archived logs can be stored in more than one location (duplexing or multiple destinations), since they are vital for recovery. For production databases, it is recommended that you use the archive log feature with multiple destinations.



Multiple Archiver Processes

In earlier releases, only one archive process was permitted in each instance. This archiver process was a bottleneck in situations where there was a lot of transaction activity in the database. To overcome this problem, a database administrator had to perform archive operations from a session manually. When multiple archive destinations are specified and frequent changes are made to the database, the archiver may fall behind and cause the LGWR to halt temporarily in performing its task. To prevent such a situation from occurring, a database administrator can start multiple archiver processes.

Multiple Remote Archive Destinations

There are two ways to specify multiple archive destinations for an instance.

- Enable a database administrator to specify both local and remote locations to receive archived log files. Optionally, the destinations can be specified as either required or desirable.
If a remote destination is specified for archiving, a new process, the remote file server (RFS), receives the file at the remote site and stores it at a specified directory.
- Specify only local destinations.

Categories of Failures

Categories of Failures

- **Statement failure**
- **User process failure**
- **User error**
- **Instance failure**
- **Media failure**

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE®**

Categories of Failures

Different types of failures may occur in an Oracle database environment. These include:

- Statement failure
- User process failure
- User error
- Instance failure
- Media failure

Each type of failure requires a varying level of involvement by the DBA to recover effectively from the situation. In some cases, recovery depends on the type of backup strategy that has been implemented. For example, a statement failure requires little DBA intervention, whereas a media failure requires the DBA to employ a tested recovery strategy.

Common Causes of Statement Failure

Causes of Statement Failures

- **Logic error in an application**
- **Attempt to enter bad data into the table**
- **Attempt an operation with insufficient privileges**
- **Attempt to create a table but exceed allotted quota limits**
- **Attempt an INSERT or UPDATE to a table, causing an extent to be allocated, but with insufficient free space left in the tablespace**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Statement Failure

Statement failure occurs where there is a logical failure in the handling of a statement in an Oracle program. Types of statement failures include:

- A logical error occurs in the application.
- The user attempts to enter bad data into the table, perhaps violating integrity constraints.
- The user attempts an operation with insufficient privileges, such as an insert on a table using only SELECT privileges.
- The user tries to create a table but exceeds the user's allotted quota limit.
- The user attempted an INSERT or UPDATE on a table, causing an extent to be allocated, but with insufficient free space in the tablespace.

Note: When a statement failure is encountered, it is likely that the Oracle server or the operating system will return an error code and a message. The failed SQL statement is automatically rolled back, then control is returned to the user program. The application developer or DBA can use the Oracle error codes to diagnose and help resolve the failure.

Resolutions for Statement Failures

Resolutions for Statement Failures

- **Correct the logic flow of the program.**
- **Modify and reissue the SQL statement.**
- **Provide the necessary database privileges.**
- **Change the user's quota limit by using the ALTER USER command.**
- **Add file space to the tablespace.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Statement Failure Resolution

DBA intervention of statement failures will vary in degree depending on the type of failure.

- Fix the application so that logical flow runs correctly. Depending on your environment this may be more of an application developer task rather than a DBA task.
- Modify the SQL statement and reissue it. This will also be more of an application developer rather than a DBA task.
- A DBA may have to provide the necessary database privileges for the user to complete the statement successfully.
- A DBA may have to issue the “alter user” command to change the quota limit.
- A DBA may have to add file space to the tablespace. Technically, the DBA should make sure this does not happen; however, in some cases it may be necessary to add file space. A DBA can also use the RESIZE and AUTOEXTEND for data files.

Causes of User Process Failures

Causes of User Process Failures

- The user performed an abnormal disconnect in the session.
- The user's session was abnormally terminated.
- The user's program raised an address exception terminating the session.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Causes of User Process Failures

A user's process may fail for a number of reasons; however, the more common causes include:

- The user performed an abnormal disconnect in the session. For example, a user issues a [Ctrl] + [Break] in SQL*Plus while connected to a database in a client-server configuration.
- The user's session was abnormally terminated. One possible scenario is the user rebooted the client while connected to a database in a client-server configuration.
- The user's program raised an address exception terminating the session. This is common if the application does not properly handle exceptions when they are raised.

Resolving User Process Failures

Resolution of User Process Failures

- PMON rolls back the transaction and releases any resources and locks being held by it.
- The PMON process detects an abnormally terminated user process.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

User Process Failure and DBA Action

The DBA will rarely need to take action to resolve user process errors. The user process cannot continue to work, although the Oracle server and other user processes will have minimal impact on the system or other users.


PMON Background Process

The PMON background process is usually sufficient for cleaning up after an abnormally terminated user process.


- The PMON process detects an abnormally terminated server process.
- The PMON process rolls back the transaction of the abnormally terminated process, and releases any resources and locks it has acquired.

Possible User Error Failures


Possible User Error Failures



```
SQL> DROP TABLE Employee;
```



```
SQL> TRUNCATE TABLE Employee;
```



```
SQL> DELETE FROM Employee;
```

```
SQL> UPDATE Employee  
2 SET SALARY = SALARY * 1.5;  
SQL> COMMIT;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

User Error Failures

DBA intervention is usually required to recover from user errors.

Common Causes of User Error Failures

- The user accidentally drops or truncates a table.
- The user deleted all rows in a table that are required.
- The user committed data, but discovered an error in committed data.

Resolving User Errors

Resolution of User Errors

- Train the database users.
- Recover from a valid backup.
- Bring back a table export.
- Recover with a point-in-time recovery.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

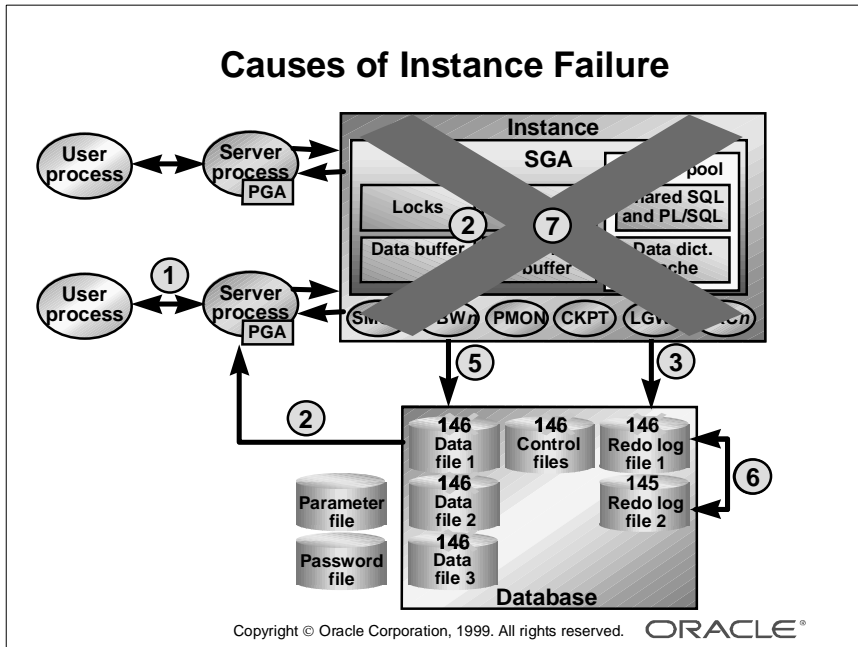
Minimizing User Error Failures

A key issue in any database and application environment is to make sure that users are properly trained and are aware of database availability and integrity implications.

A DBA should understand the types of applications and business operations that may result in loss of data from user errors and how to implement recovery measures such as recovering from a valid backup.

Some recovery situations may be quite extensive such as restoring an instance and database to a point-in-time just prior to the error, exporting the lost data, then importing that data into the database in which it was lost.

Instance Failure



Instance Failure

An instance failure may occur for numerous reasons:

- A power outage occurs that causes the server to become unavailable.
- The server becomes unavailable due to hardware problems such as a CPU failure or memory corruption or the operating system crashes.
- One of the Oracle server background processes (DBWR, LGWR, PMON, SMON, CKPT) experiences a failure.

To recover from instance failure, the DBA:

- Starts the instance by using the “startup” command. The Oracle server will automatically recover, performing both the roll forward and rollback phases.
- Investigates the cause of failure by reading the instance `alert.log` file and any other trace files that were generated during the instance failure.

Recovery from Instance Failure

Recovery from Instance Failure

- **No special recovery action needed from DBA**
- **Start the instance**
- **Wait for the database to be opened notification**
- **Notify users**
- **Check alert file to get the reason of the failure**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Instance Recovery

Instance recovery restores a database to its transaction consistent state just prior to instance failure. The Oracle server automatically performs instance recovery when the database is opened if it is necessary.

- No recovery action needs to be performed by you. All required redo information is read by SMON. To restore from this type of failure, start the database:

```
SQL> connect / as sysdba;
Connected.
SQL> startup pfile=initDB00.ora;
. . .
Database opened.
```

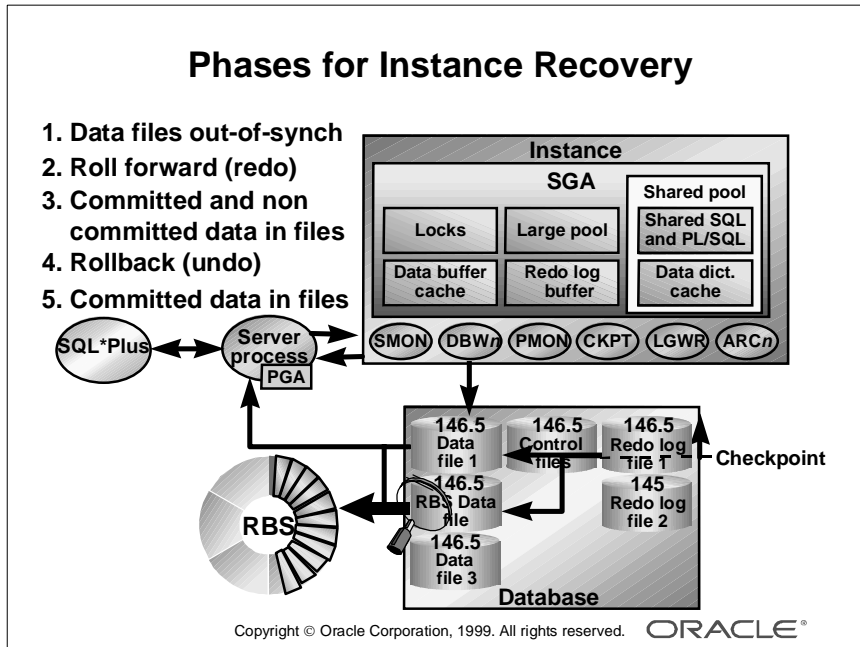
- After the database has opened, notify users that any data that they did not commit will need to be re-entered.

Instance Recovery (continued)

Note

- There may be a time delay between starting the database and the “Database opened” notification—this is the roll forward phase that takes place while the database is mounted.
 - SMON performs the roll forward process by applying changes recorded in the online redo log files from the last checkpoint.
 - Rolling forward recovers data that has not been recorded in the database files, but has been recorded in the online redo log, including the contents of rollback segments.
- Rollback can occur while the database is open, since either SMON or a server process can perform the rollback operation. This allows the database to be available for users faster.

Instance Recovery Process



Instance Recovery Phases

Phase	Explanation
1	Unsyncronized files: The Oracle server determines whether a database needs recovery when unsyncronized files are found. Instance failure can cause this to happen, such as a shutdown abort. This situation causes loss of uncommitted data because memory is not written to disk and files are not synchronized before shutdown.
2	<p>Roll forward process: DBWR writes both committed and uncommitted data to the data files. The purpose of the roll forward process is to apply all changes recorded in the log file to the data blocks.</p> <p>Note</p> <ul style="list-style-type: none"> - Rollback segments are populated during the roll-forward phase. Because redo logs store both before and after data images, a rollback segment entry is added if an uncommitted block is found in the data file and no rollback entry exists. - Redo logs are applied using log buffers. The buffers used are marked for recovery and do not participate in normal transactions until they are relinquished by the recovery process. - Redo logs are only applied to a read-only data file if a status conflict occurs (that is, the file header states the file is read-only, yet the control file recognizes it as read-write, or vice versa).
3	Committed and uncommitted data in data files: Once the roll forward phase has successfully completed, all committed data resides in the data files, although uncommitted data still might exist.
4	<p>Roll-Back Phase: To remove the uncommitted data from the files, rollback segments populated during the roll forward phase or prior to the crash are used. Blocks are rolled back when requested by either the Oracle server or a user, depending on who requests the block first.</p> <p>The database is therefore available even while rollback is running. Only those data blocks participating in rollback are not available.</p>
5	Committed data in data files: When both the roll forward and rollback phases have completed, only committed data resides on disk.
6	Synchronized data files: All data files are now synchronized.

Media Failure

Causes of Media Failures

- Head crash on a disk drive
- Physical problem in reading from or writing to database files
- File was accidentally erased

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Media Failure

Media failure involves a physical problem when reading from or writing to a file that is necessary for the database to operate. Media failure is the most serious type of failure because it usually requires DBA intervention.

Common Types of Media Related Problems

- The disk drive that held one of the database files experienced a head crash.
- There is a physical problem reading from or writing to the files needed for normal database operation.
- A file was accidentally erased.

Media Failure Resolution

Resolutions for Media Failures

- The recovery strategy depends on which backup method was chosen and which files are affected.
- If available, apply archived redo log files to recover data committed since the last backup.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Media Failure Resolution

A tested backup strategy is the key component to resolving media failure problems. The ability of the DBA to minimize downtime and data loss as a result of media failure depend on the type of backups that are available. A recovery strategy, therefore, depends on the following:

- The backup method you choose and which files are affected.
- If archiving is used, you can apply archived redo log files to recover committed data since the last backup.

Database Synchronization

Database Synchronization

- All data files (except offline and read-only) must be synchronized for the database to open.
- Synchronization is based on the current checkpoint number.
- Applying redo logs synchronizes files.
- Redo logs are automatically requested by the Oracle server.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

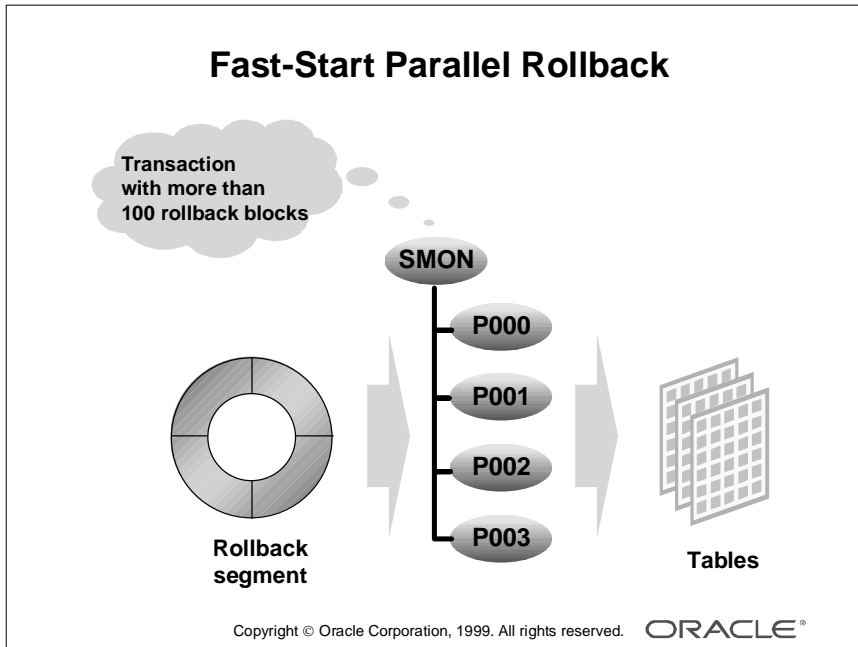
Database Synchronization

An Oracle database cannot be opened unless all data files, redo logs, and control files are synchronized. In this case, recovery is required.

Database File Synchronization

- For the database to open, all data files must have the same checkpoint number, unless they are offline or part of a read-only tablespace.
- Synchronization of all Oracle files is based on the current redo log checkpoint and sequence numbers.
- Archived and online redo log files recover committed transactions and roll back uncommitted transactions to synchronize the database files.
- Archived and online redo log files are automatically requested by the Oracle server during the recovery phase. Make sure logs exist in the requested location.

Fast-Start Parallel Rollback



Deferred Transaction Rollback

Deferred rollback was introduced in Oracle7.3.1. It enables a database to be opened as soon as the roll forward using the redo log files is completed. The rollback of any uncommitted transactions is done after opening the database to users.

The rollback could be done in one of the following ways:

- By SMON, which periodically scans all the rollback segments and rolls back aborted transactions as needed.
- By user processes, whereby a user process encounters a row lock held by a dead transaction and recovers the transaction, which frees the row lock, after which the user process continues processing.

Prior to Oracle8i

Deferred rollback could take a long time when parallel transactions are terminated by a system crash, because rollback is predominantly a serial operation. For example, a parallel update transaction with a degree of parallelism of 6 runs for 10 minutes, but fails due to a system crash. Rolling back this transaction serially by SMON could take one hour or more.

With Oracle8i

Fast start parallel rollback in Oracle8i enables a SMON to use parallel query slaves to complete the rollback operation. Parallel rollback is automatically started when SMON determines that the dead transaction had generated a large number of rollback blocks. The current threshold for determining whether a transaction is large is 100 rollback blocks.

Controlling Fast-Start Parallel Rollback

Define dynamic parameter

FAST_START_PARALLEL_ROLLBACK

Value	Maximum Parallel Recovery Servers
FALSE	None
LOW	2 * CPU_COUNT
HIGH	4 * CPU_COUNT

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Monitoring Parallel Rollback

V\$FAST_START_SERVERS

- **STATE:** recovering or idle
- **PID, UNDOBLKSDONE**

V\$FAST_START_TRANSACTIONS

- **USN, SLT, SEQ:** Transaction ID
- **UNDOBLKSDONE**
- **UNDOBLKSTOTAL**
- **CPUTIME:** Time in seconds

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE®**

Initialization Parameter

Fast start parallel rollback is initiated by the dynamic initialization parameter `FAST_START_PARALLEL_ROLLBACK`. The valid values for this parameter and its impact on fast start parallel rollback are shown in the table.

Dynamic Performance Views

Two new dynamic performance views are available to monitor the progress of fast start parallel rollback and the processes used in performing the rollback.

Parallel Query Processes Use the following query to monitor the use of parallel query slaves for fast start parallel rollback:

```
SELECT * FROM v$fast_start_servers;
```

STATE	UNDOBLOCKSDONE	PID
-----	-----	----
RECOVERING	99	10
IDLE	0	11
IDLE	0	12
IDLE	0	13

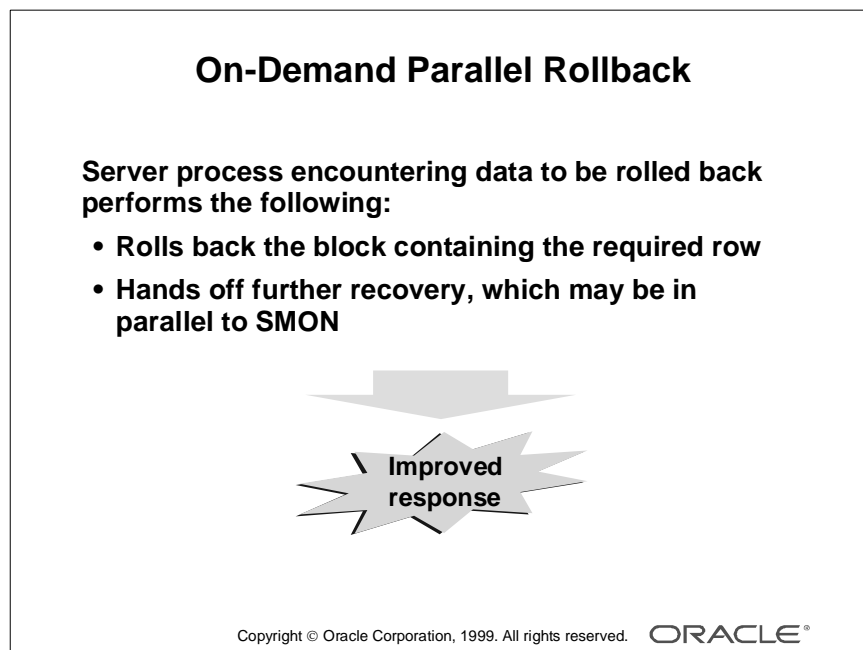
Transactions Rolled Back Use the following query to verify the status of fast start rollback:

```
SELECT usn, state, undoblksdone, undoblkstotal
FROM v$fast_start_transactions;
```

USN	STATE	UNDOBLOCKSDONE	UNDOBLOCKSTOTAL
---	-----	-----	-----
2	RECOVERING	82	365

The USN column specifies which rollback segment the rollback is taking place from, while the UNDOBLKSDONE and UNDOBLKSTOTAL indicate the amount of work done and the total amount of work, respectively.

On-Demand Parallel Rollback



On-Demand Parallel Rollback

Fast-start parallel rollback speeds up the rolling back of transactions by SMON. This does not fix the problem of a user transaction having to wait after initiating recovery of large transactions. This problem could occur when they detect locks held by a dead transaction that SMON has not yet rolled back. On-demand parallel rollback solves this problem.

In on-demand block level recovery, user transactions initiate rollback on only the block the user transaction is attempting to access. The remainder of the blocks are recovered in the background by SMON, potentially in parallel.

Quick Reference

Context	Reference
Parameters	DBWR_IO_SLAVES CONTROL_FILES FAST_START_IO_TARGET DB_BLOCK_MAX_DIRTY_TARGET LOG_CHECKPOINT_INTERVAL LOG_CHECKPOINT_TIMEOUT LOG_CHECKPOINTS_TO_ALERT DB_BLOCK_BUFFERS DB_BLOCK_SIZE LOG_BUFFER LARGE_POOL_SIZE BACKUP_TAPE_IO_SLAVES
Dynamic performance views	V\$SGA V\$SGASTAT V\$INSTANCE V\$PROCESS V\$DATABASE V\$DATAFILE V\$LOG V\$LOGFILE V\$LOG_HISTORY V\$PARAMETER V\$CONTROLFILE V\$FAST_START_SERVERS V\$FAST_START_TRANSACTIONS V\$INSTANCE_RECOVERY
Data dictionary views	none
Commands	ALTER DATABASE ADD LOGFILE ALTER DATABASE DROP LOGFILE ALTER DATABASE RENAME FILE ALTER DATABASE ADD LOGFILE MEMBER ALTER DATABASE DROP LOGFILE MEMBER ALTER SYSTEM SWITCH LOGFILE

Summary

Summary

In this lesson, you should have learned how to:

- **Identify components of instance and database with significance to recovery**
- **Configure and size the large pool**
- **Multiplex redo logs and control files**
- **Enable ARCHIVELOG mode, the ARC*n* processes, and use multiple destinations for archiving**

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Summary

- Identify types of failures and media/instance recovery
- Identify deferred transaction recovery

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Oracle Backup and Recovery Configuration

Objectives

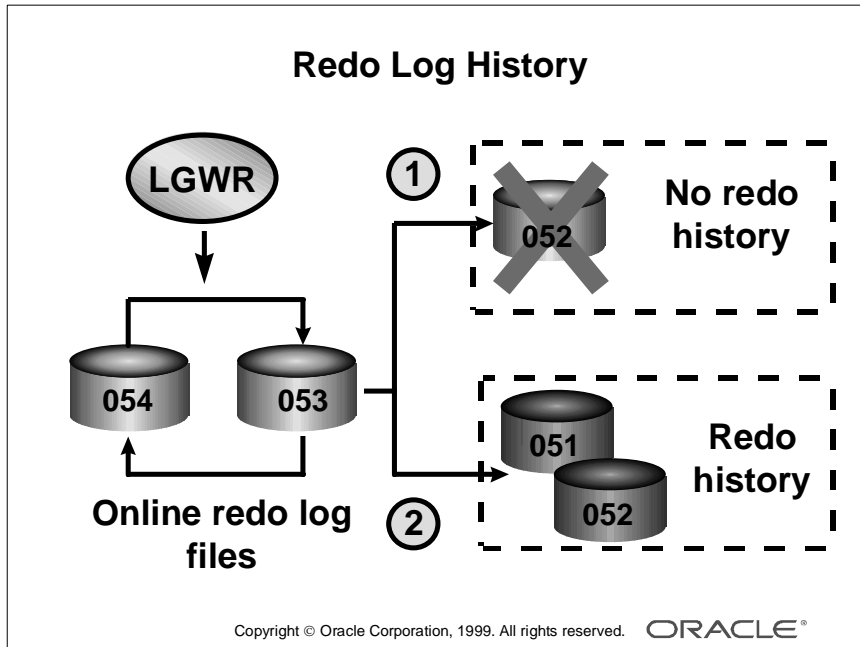
Objectives

After completing this lesson, you should be able to do the following:

- **Identify recovery implications of operating in NOARCHIVE mode**
- **Describe the differences between ARCHIVELOG and NOARCHIVELOG modes**
- **Configure a database for ARCHIVELOG mode and automatic archiving**
- **Use `init.ora` parameters to configure multiple destinations for archived log files and multiple archive processes**
- **Perform manual archive of logs**

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Redo Log

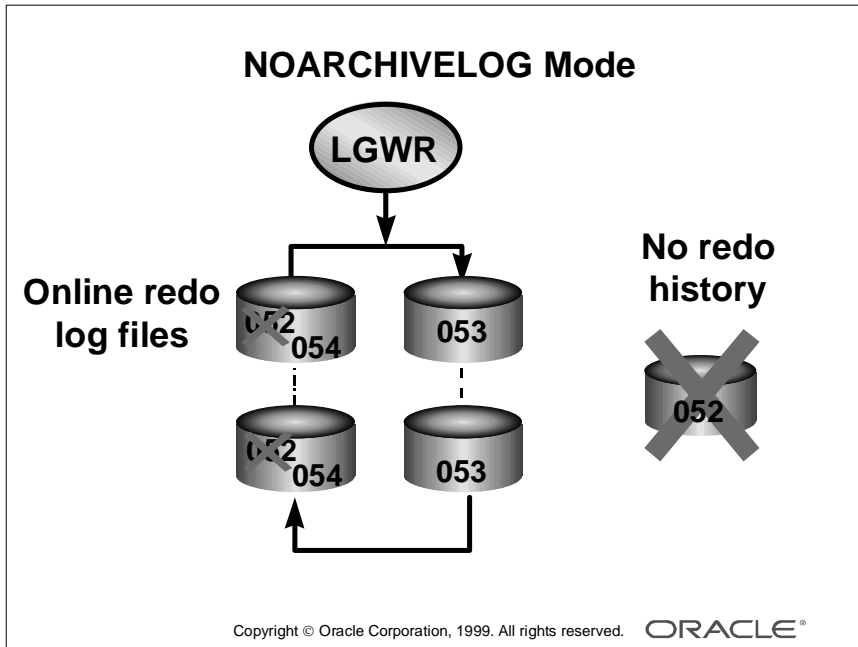


Redo Log History

Under typical database operations, all transactions are recorded in the online redo log files. This allows for automatic recovery of transactions in the event of a database failure.

- If the database is configured for NOARCHIVELOG mode, no redo history is saved to archived log files, and recovery operations are limited and a loss of transactions may occur. This is the result of the automatic recycling of log files, where older log files needed for recovery are overwritten and only the most recent part of the transaction history is available.
- You can configure a database in ARCHIVELOG mode, so that a history of redo information is maintained in archived files. The archived redo log files can be used for media recovery.
- The database can be initially created in ARCHIVELOG, but it is configured for NOARCHIVELOG mode by default.

NOARCHIVELOG Mode



NOARCHIVELOG Mode

By default, a database is created in NOARCHIVELOG mode. The characteristics of running a database in NOARCHIVELOG mode are as follows:

- Redo log files are used in a circular fashion.
- A redo log file can be reused immediately after a checkpoint has taken place.
- Once redo logs are overwritten, media recovery is only possible to the last full backup.

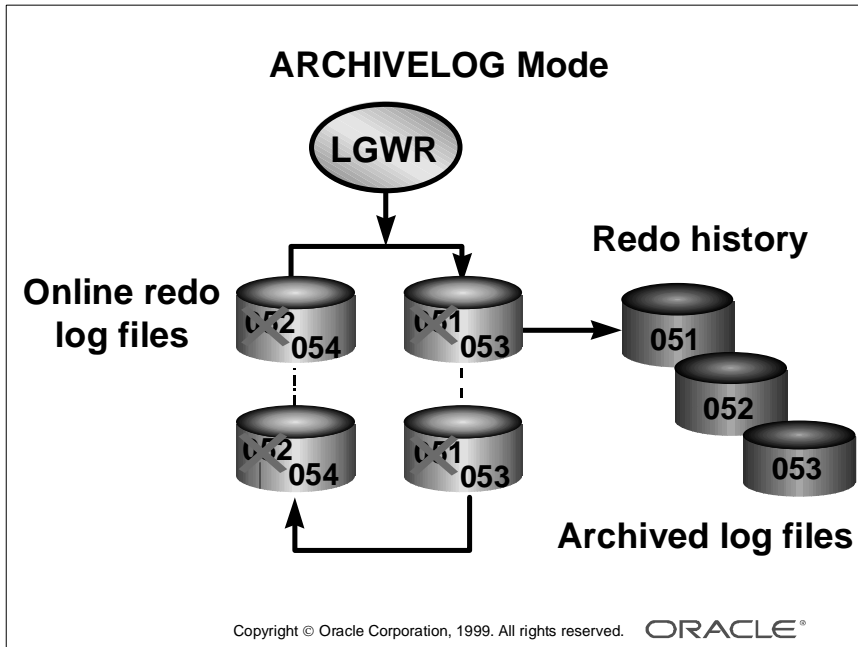
Implications of NOARCHIVELOG Mode

- If a tablespace becomes unavailable because of a failure, you cannot continue to operate the database until the tablespace has been dropped or the entire database has been restored from backups.
- You may only perform operating system backups of the database when the database is shut down.
- You must back up the entire set of database, redo, and control files during each backup.
- You will lose all data since the last full backup.
- You cannot perform online backups.

Media Recovery Options in NOARCHIVELOG mode

- You must restore the data files, redo log files, and control files from an earlier copy of a full database backup.
- If you used the Export utility to back up the database, you can use the Import utility to restore lost data. However, this results in an incomplete recovery and transactions may be lost.

ARCHIVELOG Mode



ARCHIVELOG Mode

- A filled redo log file *cannot* be reused until a checkpoint has taken place and the redo log file has been backed up by the ARC n background processes. An entry in the control file records the log sequence number of the archived log file in the log history of the control file.
- The most recent changes to the database are available at any time for instance recovery, and the archived redo log file copies can be used for media recovery.

Archiving Requirements

- The database must be in ARCHIVELOG mode. Issuing the command to put the database into ARCHIVELOG mode updates the control file. The ARC n background processes can be enabled to implement automatic archiving.
- Sufficient resources should be available to hold generated archived redo log files.

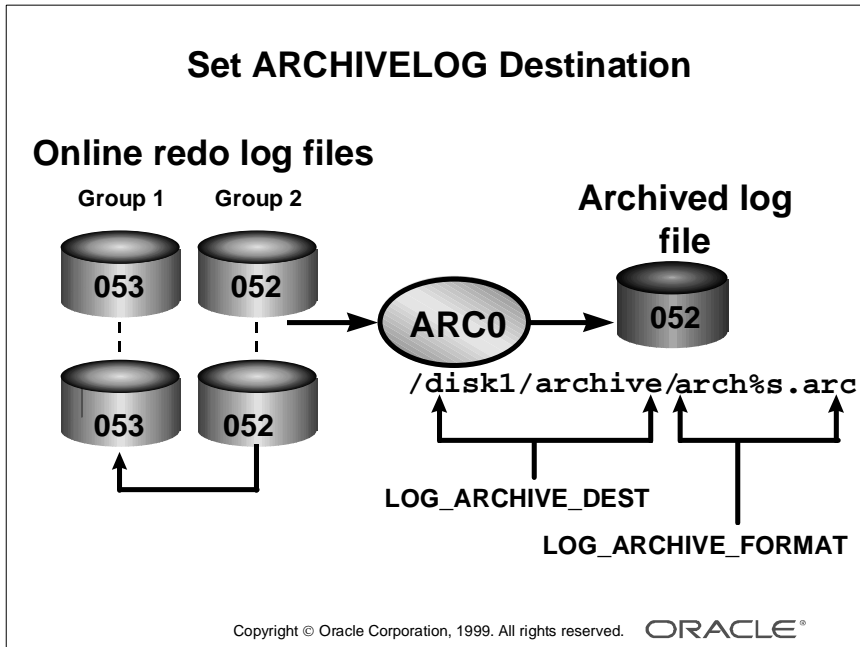
Implications of Setting ARCHIVELOG Mode in the Control File

- The database is protected from loss of data when a media failure occurs.
- You can back up the database while it is still online.
- When a tablespace other than SYSTEM goes offline as a result of media failure, the remainder of the database remains available because tablespaces (other than SYSTEM) can be recovered while the database is open.
- More online redo log groups guarantee that the archiving of online redo files can be accomplished before they are needed for reuse.

Media Recovery Options

- You can restore a backup copy of the damaged files and use archived log files to bring the datafile up-to-date while the database is online or offline.
- You can restore the database to a specific point-in-time.
- You can restore the database to the end of a specified archived log file.
- You can restore the database to a specific system change number (SCN).

Set Archivelog Destination



Set Archivelog Destination

Configure archiving by modifying the following `init.ora` parameters:

`LOG_ARCHIVE_DEST = filename`

where: *filename* Is a text string representing the default location of archived log files and may include a directory or path.

Example on UNIX: `LOG_ARCHIVE_DEST=/disk1/archive/`

Example on NT: `LOG_ARCHIVE_DEST=c:\archive\`

`LOG_ARCHIVE_FORMAT = extension`

where: *extension* Should include the variables `%s` or `%S` for log sequence number. The default value is operating system specific.

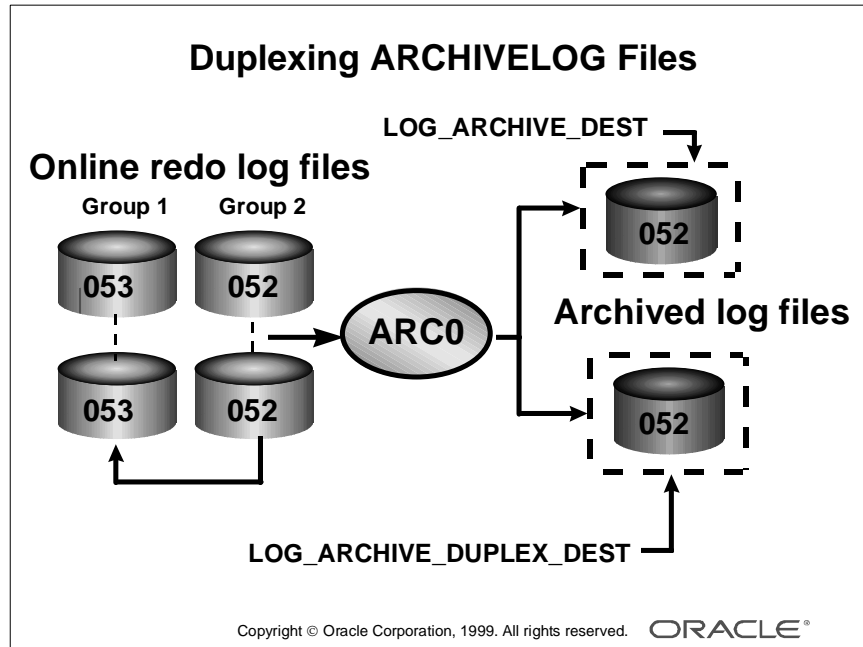
Example on UNIX and NT: `LOG_ARCHIVE_FORMAT=arch%s.arc`

Filename Options

- `%s` or `%S`: Includes the log sequence number as part of the filename.
- `%t` or `%T`: Includes the thread number as part of the filename.
- Using an uppercase `%S` causes the value to be a fixed length padded to the left with zeros.

Note: The parameter `LOG_ARCHIVE_DEST` should not refer to a raw device.

Duplexing Archived Log Files



Duplexing Archivelog Files

You can further protect archived log files against media failure by duplexing them. When duplexing is configured, copies of archived log files are written to a destination specified in the `init.ora` parameter file:

`LOG_ARCHIVE_DUPLEX_DEST = filename or device name`

where:	<i>filename</i>	Should include the variables %s or %S for log sequence number. The default value is operating system specific.
	<i>device name</i>	Is a text string to denote the default location of duplexed archive log files; may include a directory or path.

Note: The `LOG_ARCHIVE_DUPLEX_DEST` parameter can be dynamically set by using the `ALTER SYSTEM` command.

The parameter `LOG_ARCHIVE_DUPLEX_DEST` should not point to a raw device.

Specifying Multiple Archive Locations

Specifying Multiple Archive Locations

- Specify up to five archival destinations by using **LOG_ARCHIVE_DEST_***n*

Either local disk or remote database

```
log_archive_dest_1 = "LOCATION=/archive1"  
log_archive_dest_2 = "SERVICE=standby_db1"
```

- Use **LOG_ARCHIVE_DEST** and **LOG_ARCHIVE_DUPLEX_DEST**

```
log_archive_dest = /archive1/arch  
log_archive_duplex_dest = /archive2/arch
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

LOG_ARCHIVE_DEST_

n Parameters

- The LOG_ARCHIVE_DEST_
n are dynamic parameters that can be modified at the session level. A maximum of five destinations can be specified by using a suffix ranging from 1 to 5.- The destination can be either
 - A local directory, defined by using the keyword LOCATION: the location specified must be valid and cannot be an NFS-mounted directory.
 - A Net8 alias for a remote database, specified by using the SERVICE keyword: the service name specified is resolved by using the local TNSNAMES .ORA file to identify the remote instance. The initial release of Oracle8i only supports shipping of archive log files to a remote node over TCP/IP. Only one archive destination per remote database can be specified.

LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST Parameters

An alternative means of defining multiple archiving locations is to specify a primary location by using the LOG_ARCHIVE_DEST parameter and to use the LOG_ARCHIVE_DUPLEX_DEST parameter to define a backup destination. The use of this method is equivalent to using LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_DEST_2 parameters, with the exception that this method cannot be used to archive to a remote location.

The two methods for defining archive destinations are mutually exclusive. LOG_ARCHIVE_DEST_

n is recommended.

Multiple Archive Options

Multiple Archive Options

- Set archive location as **MANDATORY** or **OPTIONAL**
- Define time before retry in case of failures

```
log_archive_dest_1="LOCATION=/archive  
MANDATORY REOPEN"  
log_archive_dest_2="SERVICE=standby_db1  
MANDATORY REOPEN=600"  
log_archive_dest_3="LOCATION=/archive2  
OPTIONAL"
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

MANDATORY Versus OPTIONAL

- When using the LOG_ARCHIVE_DEST_ *n* parameters, a destination can be designated as either mandatory or optional as shown below:
 - MANDATORY implies that archiving to this destination must complete successfully before an online redo log file can be overwritten.
 - OPTIONAL implies that an online redo log file can be reused even if it has not been successfully archived to this destination.
- The default value for this option for an archive destination is OPTIONAL. There should be at least one local mandatory destination.
- If you are using the second method, LOG_ARCHIVE_DEST is implicitly considered a mandatory location, while the LOG_ARCHIVE_DUPLEX_DEST is considered an optional location.

REOPEN Attribute

- The REOPEN attribute defines whether archiving to a destination must be re-attempted in case of failure. If a number is specified along with the keyword REOPEN, as in REOPEN=600, the archiver attempts to write to this destination after the specified number of seconds following a failure. The default is 300 seconds. There is no limit on the number of attempts made to archive to a destination. Any errors in archiving are reported in the alert file at the primary site.
- If REOPEN is not specified, errors at optional destinations are recorded and ignored. No further redo logs will be sent to these destinations. Errors at mandatory destinations will prevent reuse of the online redo log until the archiving is successful. The status of an archive destination is set to ERROR whenever archiving is unsuccessful.

Specifying Minimum Number of Local Destinations

Specifying Minimum Number of Local Destinations

- **LOG_ARCHIVE_MIN_SUCCEED_DEST** parameter

```
log_archive_min_succeed_dest = 2
```

- An online redo log group can be reused only if:
 - Archiving has been done to all mandatory locations
 - The number of local locations archived is greater than or equal to the value of the **LOG_ARCHIVE_MIN_SUCCEED_DEST** parameter

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

LOG_ARCHIVE_MIN_SUCCEED_DEST and Mandatory

The number of destinations that need to be archived successfully before an online redo log file can be used is determined based on the following settings:

- The number of destinations defined as MANDATORY
- The value of the **LOG_ARCHIVE_MIN_SUCCEED_DEST** parameter

The value of **LOG_ARCHIVE_MIN_SUCCEED_DEST** parameter specifies a lower bound on the number of local destinations that need to be archived. If this number is less than the number of mandatory local destinations, it has no effect on the archiving behavior. If this number exceeds the number of mandatory local destinations, the number of local destinations archived must be at least equal to this value before an online redo log file can be reused.

Note: When configuring archiving, make sure that the path names are set according to the operating system environment. These differ between UNIX and NT.

LOG_ARCHIVE_MIN_SUCCEED_DEST and Mandatory (continued)

Example: Consider a case where LOG_ARCHIVE_MIN_SUCCEED_DEST is set to 2. If the number of mandatory local destinations is 3, then these three locations must be archived before an online redo log file can be reused. On the other hand, if the number of mandatory local archive destinations is 1, then at least one optional local archive destination must be archived before an online redo log file can be reused. In other words, the LOG_ARCHIVE_MIN_SUCCEED_DEST can be used to make archiving to one or more optional destinations mandatory, but not vice versa.

Controlling Archiving to a Destination

Controlling Archiving to a Destination

- An archival destination may be disabled by a new (dynamic) initialization parameter:
LOG_ARCHIVE_DEST_STATE _n

```
log_archive_dest_state_2 = DEFER  
log_archive_dest_state_3 = DEFER
```

- Archiving to a destination can be enabled again:

```
log_archive_dest_state_2 = ENABLE
```

```
ALTER SYSTEM SET log_archive_dest_state_3 =  
ENABLE
```

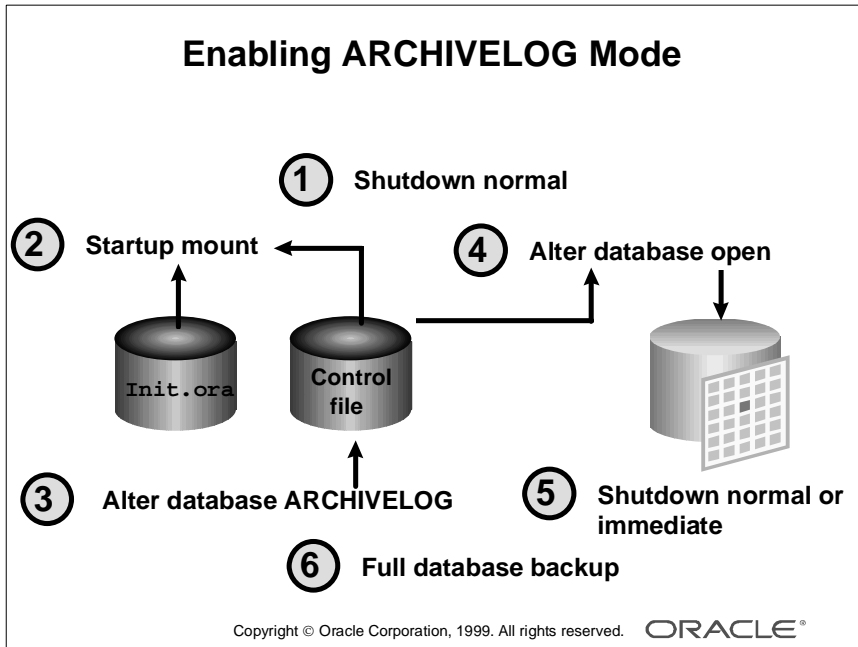
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

LOG_ARCHIVE_DEST_STATE_n Parameter

- The state of an archive destination may be changed dynamically. By default, an archive destination is in the ENABLE state, indicating that the Oracle server can use this destination.
- The state of an archive destination can be modified by setting the corresponding LOG_ARCHIVE_DEST_STATE_n parameter. For example, to stop archiving to a mandatory location temporarily when an error has occurred, the state of that destination can be set to DEFER. A destination may be defined, but set to DEFER in the parameter file. This destination may then be enabled when some other destination has an error or needs maintenance.

Note: Archiving is not performed to a destination when the state is set to DEFER. If the state of this destination is changed to ENABLE, any missed logs must be manually archived to this destination.

Enabling ARCHIVELOG Mode



Enabling ARCHIVELOG Mode

It is the task of a DBA to change the mode. The DBA changes the mode by using the ALTER DATABASE command while the database is in the MOUNT state.

```
SQL> alter database [ archivelog | noarchivelog ]
```

where: *archivelog* Establishes ARCHIVELOG mode for redo log file groups

noarchivelog Establishes NOARCHIVELOG mode for redo log file groups

Enabling ARCHIVELOG Mode (continued)

A user must have the alter system privilege to alter the archive mode of the database.

Step Number	Explanation
1	Shutdown the database: <code>SQL> shutdown immediate</code>
2	Start the database in MOUNT state to alter the archive mode of database: <code>SQL> startup mount</code>
3	Set the database in ARCHIVELOG mode by using the ALTER DATABASE command: <code>SQL> alter database archivelog;</code>
4	Open the database: <code>SQL> alter database open;</code>
5	Shut down the database: <code>SQL> shutdown immediate</code>
6	Take a full backup of database

Note: After the database mode has been changed from NOARCHIVELOG mode to ARCHIVELOG, you must back up all the database files and the control file. Your previous backup is not usable anymore because it was taken while the database was in NOARCHIVELOG mode.

The new backup taken after putting the database into ARCHIVELOG mode is the backup against which all your future archived redo log files will apply.

Setting the database in ARCHIVELOG mode *does not* enable the Automatic Archival (ARCn) processes.

Backup Manager

How to change archive methods by using Backup Manager:

Backup Manager—>Logfile—>Enable automatic archiving or logfile, disable automatic archiving in the menu.

Multiple Archive Processes

Set Multiple ARCN Processes

- The dynamic parameter controls the number of archive processes:
LOG_ARCHIVE_MAX_PROCESSES
- The parameter **LOG_ARCHIVE_START** set to **TRUE** or **FALSE** controls automatic or manual archiving

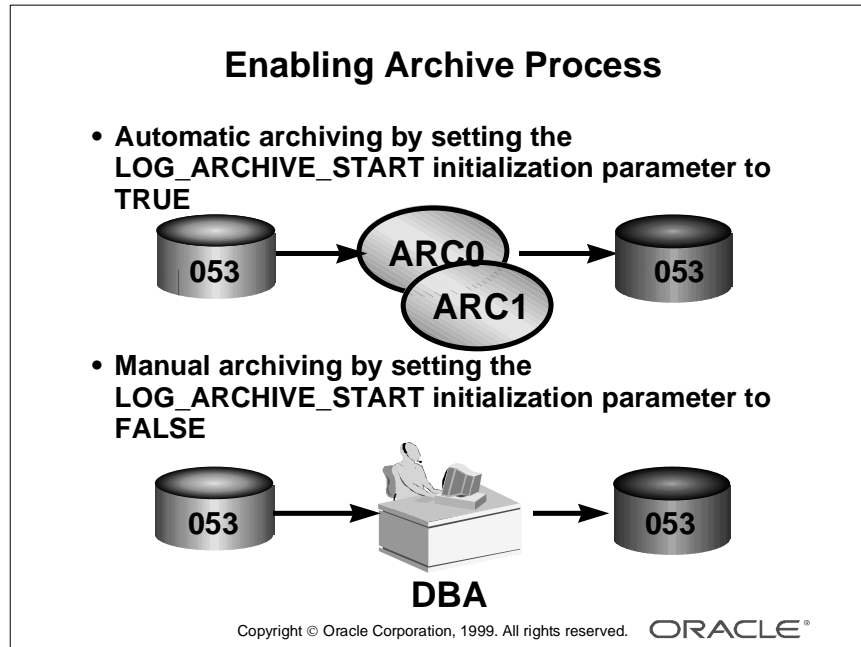
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

LOG_ARCHIVE_MAX_PROCESSES Parameter

Parallel data definition language (DDL) and parallel data manipulation language (DML) operations may generate a large amount of redo logs. A single ARC0 process to archive these redo logs might not be able to keep up. To avoid this problem, you can spawn multiple archiver processes. This can be done manually or by using a job queue.

- Oracle8i allows the database administrator to define multiple archive processes by using the LOG_ARCHIVE_MAX_PROCESSES parameter.
- When LOG_ARCHIVE_START is set to TRUE, an Oracle instance starts up with as many archiver processes as defined by LOG_ARCHIVE_MAX_PROCESSES.
- A maximum of ten archive processes are allowed. The minimum value is one.
- The DBA can always spawn additional archive processes or kill some superfluous archive processes at any time during the instance life.

Enabling Archive Process



The Archive Process

Once a database is set in ARCHIVELOG mode, the DBA must decide whether online redo log files are to be archived automatically or manually. This is the second step in getting archive logs created to use for later recoverability.

Automatic Versus Manual Archiving

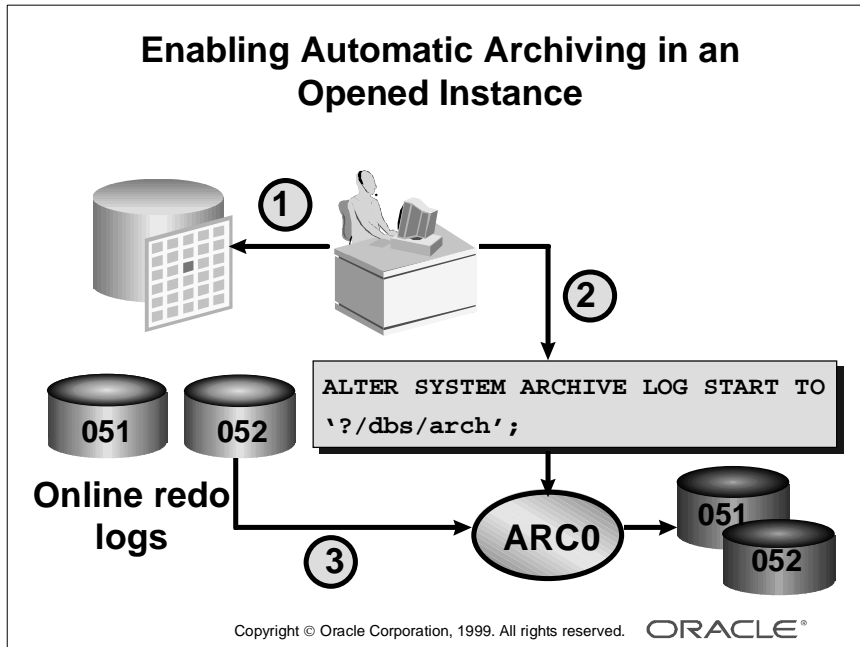
- In automatic archiving, the ARC_n background processes are enabled and they copy redo log files as they are filled.
- In manual archiving, the DBA must use SQL*Plus, SQL Worksheet, or Backup Manager.
- It is recommended that you enable automatic archival log files.

Guidelines

- Before deciding on the archive mode (automatic or manual), you must set the database in ARCHIVELOG mode.
- Failure to switch to ARCHIVELOG mode will prevent ARC*n* from copying redo log files.
- The database should be shut down cleanly (by using the normal, immediate, or transactional option) before enabling the archive process.

Note: If the archive processes (ARC*n*) fail for any reason, once transaction activity has filled up all the redo logs, the Oracle server hangs. This is a legitimate hang, because setting the database in ARCHIVELOG mode tells the Oracle server not to overwrite the online redo logs unless it is archived. Thus the archiving of online redo logs must keep pace with the transaction activity on the system (generation of redo logs).

Enabling the Archive Process in an Open Instance



The Archive Process in an Opened Instance

You can enable the ARC_n processes in an opened instance, if it is not done through the initialization parameter `LOG_ARCHIVE_START`. The database should be in ARCHIVELOG mode.

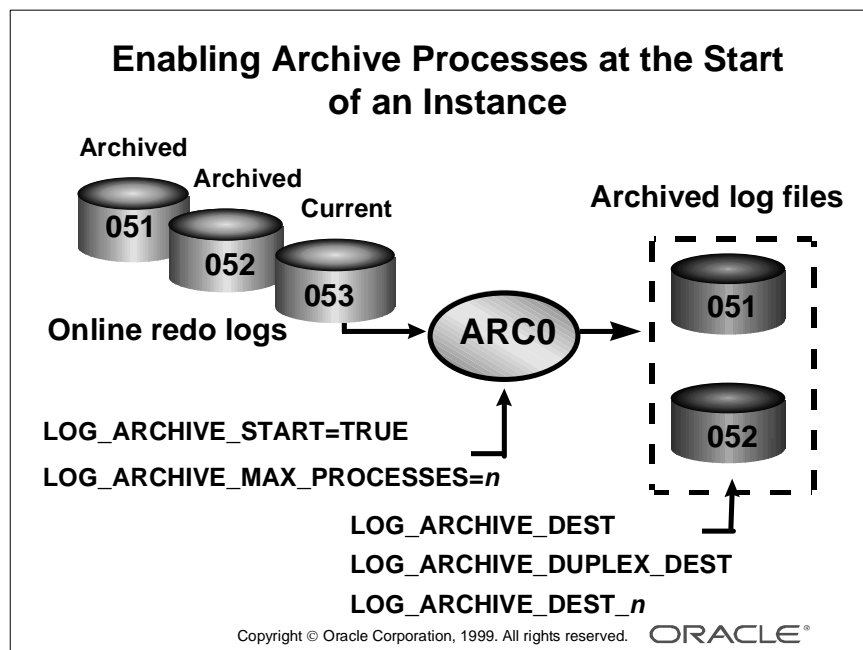
The Archive Process in an Opened Instance (continued)

Step	Explanation
1	Open the database. Make sure that database is in ARCHIVELOG mode. SQL> archive log list;
2	Enable the Archiver processes (ARC <i>n</i>) to archive log files automatically. UNIX: SQL> alter system archive log start to '/u04/Oracle/TEST/log'; NT: SQL> alter system archive log start to 'c:\u04\Oracle\TEST\log';
3	The ARC <i>n</i> processes automatically archive log files as they are filled.

Characteristics

- If the ARC*n* processes are not initiated through initialization parameter file, then you must restart the ARC*n* processes each time you restart the instance.
- By default, ARC*n* is disabled.

Enabling Archive Processes at Start of Instance



Archive Processes at the Start of an Instance

If the database is in ARCHIVELOG mode, then Archive processes can be started every time the database instance is started by setting the `init.ora` parameter file:

`LOG_ARCHIVE_START = boolean`

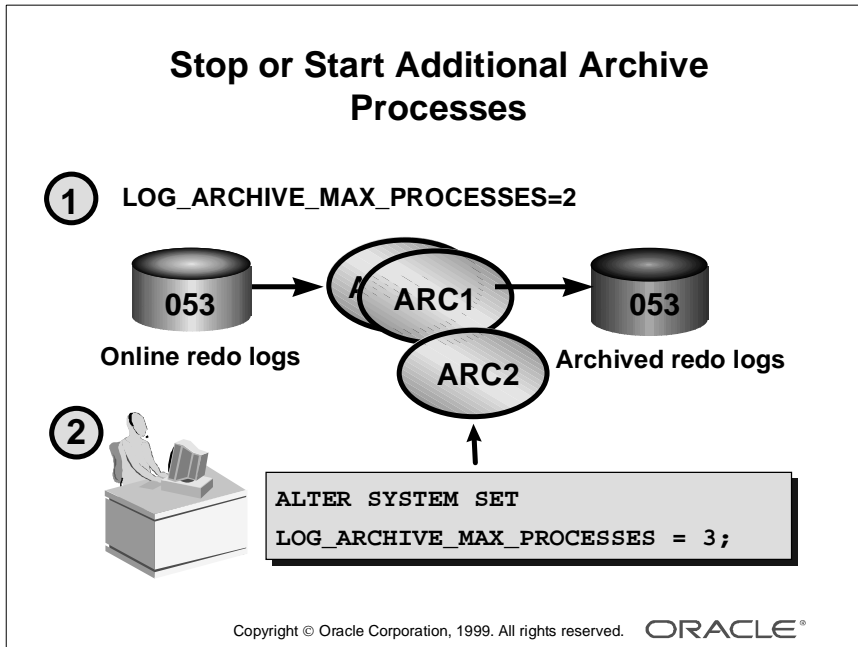
where: *boolean*

TRUE automatically starts the `ARCn` processes upon instance startup depending on the value of `LOG_ARCHIVE_MAX_PROCESSES`.

FALSE inhibits `ARCn` from starting upon instance startup.

Once the `init.ora` parameter file is set, the `ARCn` processes automatically start upon instance startup, eliminating the need for the DBA to start automatic archiving manually.

Stop or Start Additional Archive Processes



Dynamic Number of ARC_n Processes

During heavy transactional load or activity, the DBA can temporarily start additional archive processes to prevent bottlenecks on the archiving workload. Once the transactional activity comes down to a normal level, the DBA can stop some ARC_n processes.

For example, every day of the month, the database starts up with two archive processes. During the last day of each month, the activity always increases:

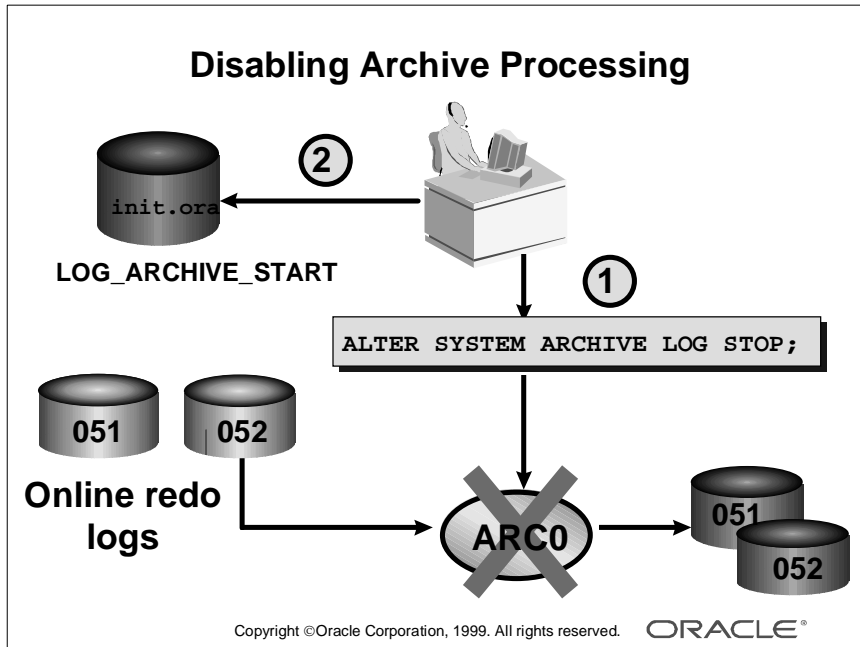
```
SQL> alter system set LOG_ARCHIVE_MAX_PROCESSES=3;
```

The day after, if the database is not shut down, the DBA can issue the following SQL command so as to stop the additional archive process:

```
SQL> alter system set LOG_ARCHIVE_MAX_PROCESSES=2;
```

Note: If the database is shut down at night, the next day the database would start again with only two archive processes as it is set up in the `init.ora` file.

Disabling Archive Processing



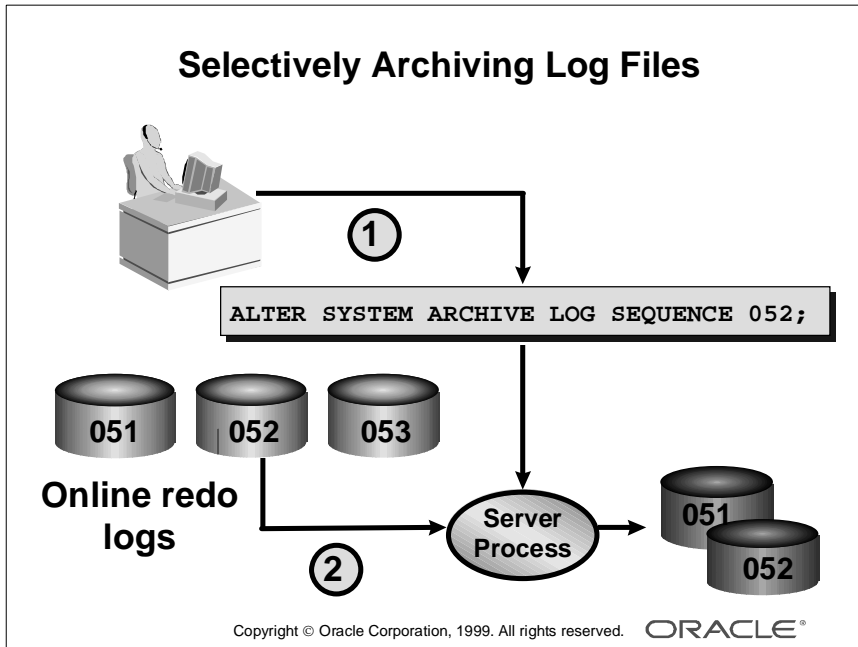
Disabling the ARC_n Processes By Using the Command Line

You can always stop archive processes regardless of how you started it by using the ALTER SYSTEM command in SQL*Plus or SQL Worksheet or by using Backup Manager.

Step	Explanation
1	Execute the command to stop the ARC _n processes, if ARC _n processes have been already enabled: SQL> alter system archive log stop;
2	Ensure that automatic archiving is not enabled upon instance startup by editing the init.ora file and setting the parameter: LOG_ARCHIVE_START=FALSE

Make sure that the database is set to NOARCHIVELOG mode before or immediately after stopping ARC_n processes. Stopping ARC_n processes does not set the database in NOARCHIVELOG mode. When all groups of redo logs are used and not archived, the database may hang if it is in ARCHIVELOG mode.

Selectively Archiving Log Files



Archiving Log Files Selectively

If you have DBA privileges, you can manually archive redo log files by using the following command:

Step	Explanation
1	Execute the alter system archive log [options] SQL command: <code>SQL> alter system archive log sequence 052;</code>
2	The server process for the user executing the command will perform the archiving of the online redo log files.


Options

When you selectively archive online redo log files you can use the following options with the alter system archive log command:

Option	Description
THREAD	Specifies thread containing the redo log file group to be archived (for Oracle Parallel Server)
SEQUENCE	Archives the online redo log file group identified by the log sequence number
CHANGE	Archives based upon the SCN
GROUP	Archives online redo log file group
CURRENT	Archives the current redo log file group of the specified thread
LOGFILE	Archives redo log file group with member identified by filename
NEXT	Archives the oldest online redo log file group that has not been archived
ALL	Archives all online redo log file groups
START	Enables automatic archiving of redo log file groups
TO	Specifies the location to which the redo log file group is archived
STOP	Disables automatic archiving of redo log file groups

Obtaining Archive Log Information

Obtaining Archive Log Information

- Data dictionary views
 -  **V\$ARCHIVED_LOG**
 - V\$ARCHIVE_DEST**
 - V\$LOG_HISTORY**
 - V\$DATABASE**
 - V\$ARCHIVE_PROCESSES**
- Command line

```
ARCHIVE LOG LIST;
```

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE®**

Dynamic Views

You can manage and view log files by using data dictionary views.

- **V\$ARCHIVED_LOG**: Displays archived log information from the control file.
- **V\$ARCHIVE_DEST**: For the current instance, describes all archive log destinations, the current value, mode, and status.

```
SELECT destination, binding, target, status
FROM v$archive_dest;
```

DESTINATION	BINDING	TARGET	STATUS
-----	-----	-----	-----
/db1/oracle/DEMO/arch	MANDATORY	PRIMARY	VALID
/db2/oracle/DEMO/arch	OPTIONAL	PRIMARY	DEFERRED
standbyDEMO	OPTIONAL	STANDBY	ERROR
	OPTIONAL	PRIMARY	INACTIVE
	OPTIONAL	PRIMARY	INACTIVE

Dynamic Views (continued)

Note: The query displays five rows, each representing information for a possible destination. A status of INACTIVE indicates that this destination is not defined. A status of VALID indicates the destination is enabled and error-free.

To check for errors and the log sequence number at which the error occurred for each destination, use the following query:

```
SELECT destination, fail_sequence, error
FROM v$archive_dest
WHERE status='ERROR';
```

DESTINATION	FAIL_SEQ	ERROR
standbyDEMO	2010	ORA-12154: TNS:could not resolve service name

1 row selected.

- **V\$LOG_HISTORY:** Contains log file information from the control file.
- **V\$DATABASE:** Current state of archiving.
- **V\$ARCHIVE_PROCESSES:** Provides information about the state of the various ARCH processes for the instance.

```
SELECT * FROM v$archive_processes;
```

PROCESS	STATUS	LOG_SEQUENCE	STAT
0	ACTIVE	2014	BUSY
1	ACTIVE	0	IDLE
2	ACTIVE	0	IDLE
3	STOPPED	0	IDLE
4	STOPPED	0	IDLE
5	STOPPED	0	IDLE
6	STOPPED	0	IDLE
7	STOPPED	0	IDLE
8	STOPPED	0	IDLE
9	STOPPED	0	IDLE

10 rows selected.

One row for each of the 10 possible archiver processes is displayed. A status of ACTIVE indicates that the process is up and running. A process that is currently archiving has a state of BUSY. The LOG_SEQUENCE column for a busy process shows the current log sequence number it is archiving.

Archive Log Information

The ARCHIVE LOG LIST command provides the DBA with information about the log mode and status of archiving for the database:

```
SQL> archive log list;
Database log mode                Archive mode
Automatic archival                Enabled
Archive destination              /oracle/backup/archive/
Oldest online log sequence       1304
Next log sequence to archive     1305
Current log sequence             1305
```

Archive List Display	Description
Database log mode	Current mode of archiving
Automatic archival	Status of the optional Archiver processes
Archive destination	Destination to which log files will be copied (either by manual instruction or the detached process); shows one of the destinations, even if they are all mandatory
Oldest online log sequence	Sequence number of oldest online log
Next log sequence to archive	Next redo log to archive (only displayed in ARCHIVELOG mode)
Current log sequence	Sequence number of current log file

Recovery Configuration

Factors Influencing Time to Recover

- Fast-start recovery time is at best an estimate
- Recovery may take longer because:
 - Checkpoint target is changed only at specific time intervals
 - Additional recovery activities such as reading logs are not accounted for
 - Recovery time may be faster if parallel recovery is used

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Recovery Time and FAST_START_IO_TARGET Parameter

The recovery time as determined by using FAST_START_IO_TARGET is only a target threshold and there are no guarantees. Blocks in cache may have been dirtied since the last time the threshold was checked. The time calculation also does not take into account other recovery activities, such as reading redo log files, and lock remastering. However, typically these activities account for less than 5% of the recovery activities.

The actual elapsed time for recovery may be less, especially if parallel recovery is being used.

Monitoring Recovery Time

Monitoring Recovery Time

- Define FAST_START_IO_TARGET based on:
 - Service level required
 - AVGIOTIM column in V\$FILESTAT
- Check impact of parameters from:
 - V\$INSTANCE_RECOVERY
 - V\$TARGETRBA

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

FAST_START_IO_TARGET Value

To specify a value for FAST_START_IO_TARGET, decide on the required service level after discussion with users. Translate this time to equivalent number of data file I/O by using the average I/O time statistic from the view V\$FILESTAT.

Monitoring Impact of Parameters on Recovery Time

V\$INSTANCE_RECOVERY View

- **RECOVERY_ESTIMATED_IOS:** The estimated number of data blocks to be processed during recovery based on the in-memory value of the fast-start checkpoint parameter
- **ACTUAL_REDO_BKLS:** The current number of redo blocks required for recovery
- **TARGET_REDO_BKLS:** The goal for the maximum number of redo blocks to be processed during recovery. This value is the minimum of the following four columns:
 - **LOG_FILE_SIZE_REDO_BKLS:** The number of redo blocks to be processed during recovery to guarantee that a log switch never has to wait for a checkpoint
 - **LOG_CHKPT_TIMEOUT_REDO_BKLS:** The number of redo blocks that need to be processed during recovery to satisfy **LOG_CHECKPOINT_TIMEOUT**
 - **LOG_CHKPT_INTERVAL_REDO_BKLS:** The number of redo blocks that need to be processed during recovery to satisfy **LOG_CHECKPOINT_INTERVAL**
 - **FAST_START_IO_TARGET_REDO_BKLS:** The number of redo blocks that need to be processed during recovery to satisfy **FAST_START_IO_TARGET**

V\$TARGETRBA View **INC_EST_RCV_READS:** Number of estimated blocks needing reading in recovery (stored in control file)

Summary

Summary

In this lesson, you should have learned that:

- Complete database recovery requires:
 - ARCHIVELOG mode
 - Archiving of redo logs
 - A database backup immediately after it is put into ARCHIVELOG mode
- Recovery time depends on:
 - FAST_START_IO_TARGET parameter
 - Parallelism
 - Other nonpredictable factors

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Quick Reference

Context	Reference
Parameters	FAST_START_IO_TARGET LOG_ARCHIVE_START LOG_ARCHIVE_MAX_PROCESSES LOG_ARCHIVE_DEST LOG_ARCHIVE_DEST_ <i>n</i> LOG_ARCHIVE_FORMAT LOG_ARCHIVE_DUPLEX_DEST LOG_ARCHIVE_MIN_SUCCEED_DEST
Dynamic performance views	V\$ARCHIVED_LOG V\$ARCHIVE_DEST V\$LOG_HISTORY V\$DATABASE V\$ARCHIVE_PROCESSES V\$FILESTAT V\$INSTANCE_RECOVERY V\$TARGETRBA
Data dictionary views	none
Commands	ALTER SYSTEM ARCHIVE [options] ALTER SYSTEM SET LOG_ARCHIVE_MAX_PROCESSES= <i>n</i> ALTER DATABASE [ARCHIVELOG NOARCHIVELOG]

Physical Backups Without Oracle Recovery Manager

Objectives

Objectives

After completing this lesson, you should be able to do the following:

- **Describe the recovery implications of closed and opened database backups**
- **Perform closed and opened database backups**
- **Identify the backup implications of the Logging and Nologging options**

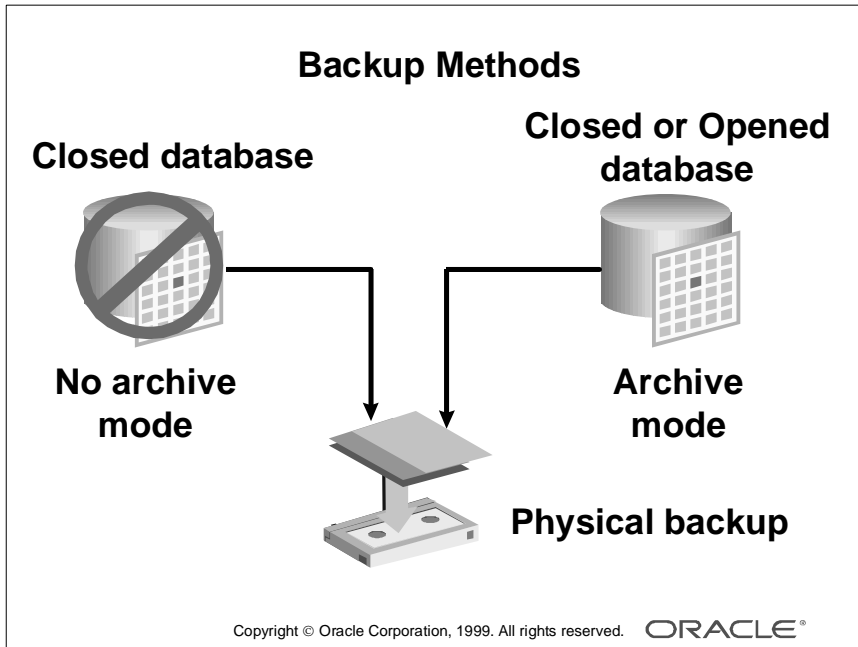
Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Objectives

- **Identify the different types of control file backups**
- **Discuss backup issues associated with read-only tablespaces**
- **List the data dictionary views useful for backup operations**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE[®]

Overview



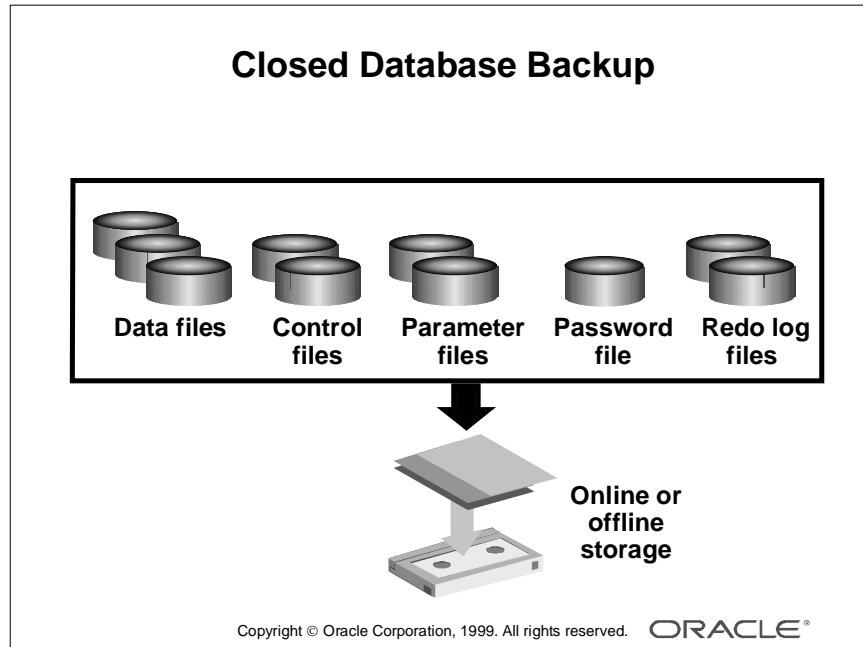
Evaluating Backup Methods

You can safeguard against loss of data resulting from media failures by choosing the most appropriate backup method for maximum data recovery. A database backup is an operating system backup of data files while the database is opened or closed.

Physical Backup Methods

- Operating system backup without archiving: Used to recover to the point of the last backup after a media failure.
- Operating system backup with archiving: Used to recover to the point of failure after a media failure.

Closed Database Backups



Closed Database Backups

A closed database backup is an operating system backup of all the data files, control files, parameter files, and the password file that constitute an Oracle database.

Define an operating system backup procedure that will always back up the Oracle data files, control files, parameter files, and the password file as part of a strategy to safeguard against potential media failures that can damage these files.

Ensure the complete pathnames of the files are noted and used appropriately in backup. In a multiple database environment, care must be taken to associate these files with the corresponding database through some naming convention, since the names of the parameter files and password files are not recorded in the dictionary.

Note: It is not necessary to include the online redo log files as part of a whole database backup, if the database has been shut down cleanly, by using a normal, transactional or immediate option. However, in cases where it is required to restore the entire database, the process is simplified.

Advantages of Closed Database Backups

- **Conceptually simple**
- **Easy to perform**
- **Require little operator interaction**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Advantages

- A closed database backup is conceptually simple because all you need to do is:
 - Shut down the database
 - Copy all required files to the backup location
 - Open the database
- A minimal number of commands are necessary to perform a closed database backup.
- You can automate the closed database backup process by executing a simple script that requires minimal operator interaction and does the following:
 - Shuts down the database
 - Copies the data files
 - Opens the database
- All files copied during a closed database backup are consistent to a point-in-time. No transactions occur because the database is unavailable for use.


Disadvantages

- For business operations where the database must be continuously available, a closed database backup is unacceptable because the database is unavailable during backup.
- The amount of time that the database is unavailable is affected by the size of the database, the number of data files, and the speed with which the copy operations on the data files can be performed. Sometimes this may not be consistent within an available window of downtime and the DBA must choose another type of backup.
- A recovery is only as good as the last full closed database backup, and lost transactions may have to be entered manually following a recovery operation.

Obtaining Database File Information

Database File Information

Data dictionary views



V\$DATAFILE
V\$CONTROLFILE
V\$LOGFILE
DBA_DATA_FILES

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Dynamic Views

You can obtain information about the files of a database by querying the V\$DATAFILE, V\$CONTROLFILE, V\$LOGFILE, and V\$TABLESPACE views.

Examples

Use the V\$DATAFILE view to obtain a listing of the names and status for all data files.

```
SQL> select name,status from v$datafile;
```

NAME	STATUS
-----	-----
/users/dba00/u01/system01.dbf	SYSTEM
/users/dba00/u03/temp01.dbf	ONLINE
/users/dba00/u03/data01_1.dbf	ONLINE

Examples (continued)

Use the V\$CONTROLFILE view to display the names of all control files.

```
SQL> select name from v$controlfile;
NAME
```

```
-----
/users/dba00/u01/cntrl1.con
/users/dba00/u02/cntrl2.con
```

Use the V\$LOGFILE view to display the names of all redo log files.

```
SQL> select member from v$logfile;
MEMBER
```

```
-----
/users/dba00/u01/log1a.rdo
/users/dba00/u02/log2a.rdo
```

Use the V\$TABLESPACE and V\$DATAFILE data dictionary views to obtain the listing of all data files and their respective tablespaces. This is very useful when setting up scripts to perform opened database backups, so you can ensure that you copy all files at the operating system level.

```
SQL> select t.name Tablespace,f.name Datafile
       2 from v$tablespace t, v$datafile f
       3 where t.ts# = f.ts#
       4 order by t.name;
```

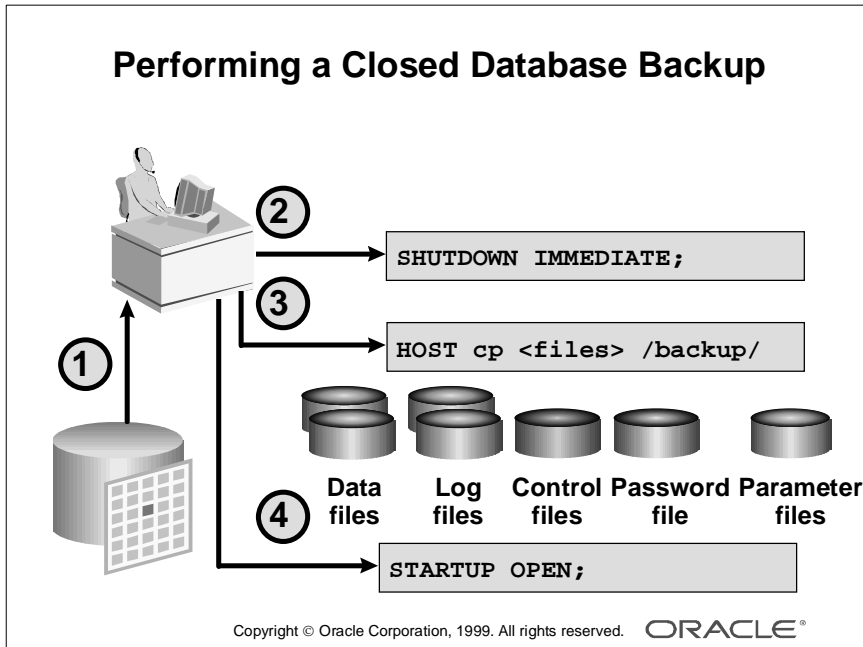
TABLESPACE	DATAFILE
INDEX_03	/users/dba00/u02/index03_3.dbf
RBS	/users/dba00/u03/rbs01.dbf
SYSTEM	/users/dba00/u01/system01.dbf
TEMP_DATA	/users/dba00/u03/temp01.dbf
USER_DATA	/users/dba00/u03/data01_1.dbf

Backup Manager

You can also use Backup Manager to identify the data file for a particular tablespace:

- 1 Log on to Backup Manager as sysdba.
- 2 Expand the Tablespaces node.
- 3 Expand the tablespace, or click the name of the tablespace.

Performing a Closed Database Backup



Performing a Closed Database Backup

Perform a full closed backup while the Oracle server instance is shut down.

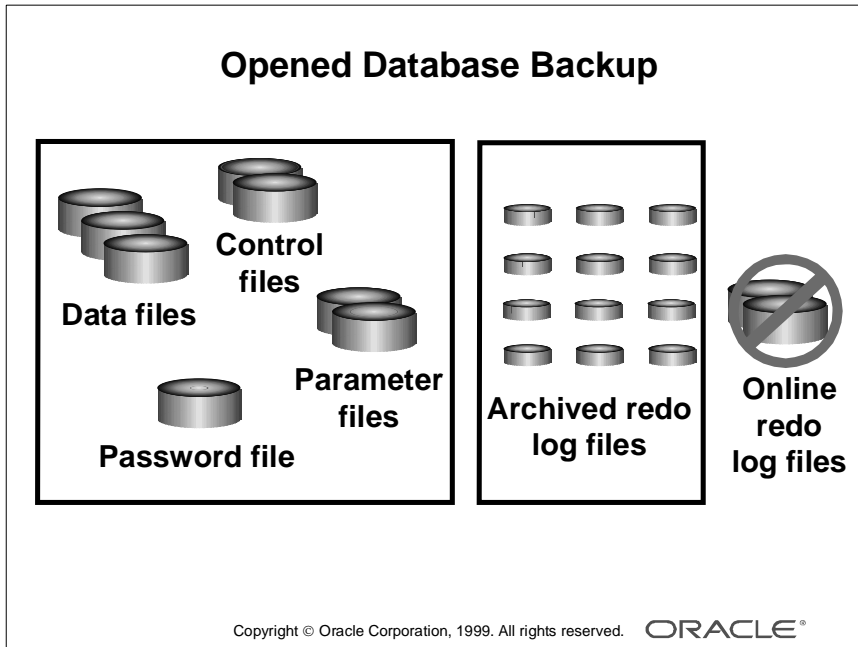
- 1** Compile an up-to-date listing of all relevant files to back up.
- 2** Shut down the Oracle instance with the shutdown normal or shutdown immediate or shutdown transactional command.
- 3** Back up all data files, redo log files, control files, parameter files, and the password file by using an operating system backup utility.
- 4** Restart the Oracle instance.

Guidelines

- The default shutdown parameter is normal. Use transactional or immediate if there is any chance that transactions or processes are still accessing the database.
- Consider a reliable, automated procedure for this operation to ensure that every file is correctly backed up.
- Back up the parameter file and the password file when performing full closed backups.
- You do not need to include files associated with read-only tablespaces in full backups.
- If the database is opened while the offline or cold backup is performed, the backup is invalid and cannot be guaranteed usable in a recovery situation.

Although the parameter file and the password file are not physically part of the database, they should be included as part of the backup.

Opened Database Backup



Opened Database Backup

Continuous-operation businesses have special implications for backup and recovery. If the business case does not allow for shutting down the database to perform backups, then there must be a mechanism to perform backups of the database while it is in use.

- A DBA can perform backups of all the tablespaces or individual data files while the database is in use, by using the opened database backup method.
- The online redo log files do not need to be backed up. You either lose the current active online redo log group while the database is still opened or you lose the current active online redo log group and the database is closed.
 - If you lose the current active online redo log group while the database is still opened, and after clearing the redo log lost, if the database crashes due to media failure before the next backup, then incomplete recovery will be required since redo information has not been archived. You should therefore immediately perform a closed whole backup.
 - If you lose the current active online redo log group and the database is closed because of a media failure, an incomplete recovery is therefore required and you will lose the transactions that were stored in the lost redo log that could not be archived yet.

Advantages of Opened Database Backups

- **Maintains high database availability**
- **Can be done at a tablespace or data file level**
- **Supports nonstop business operations**

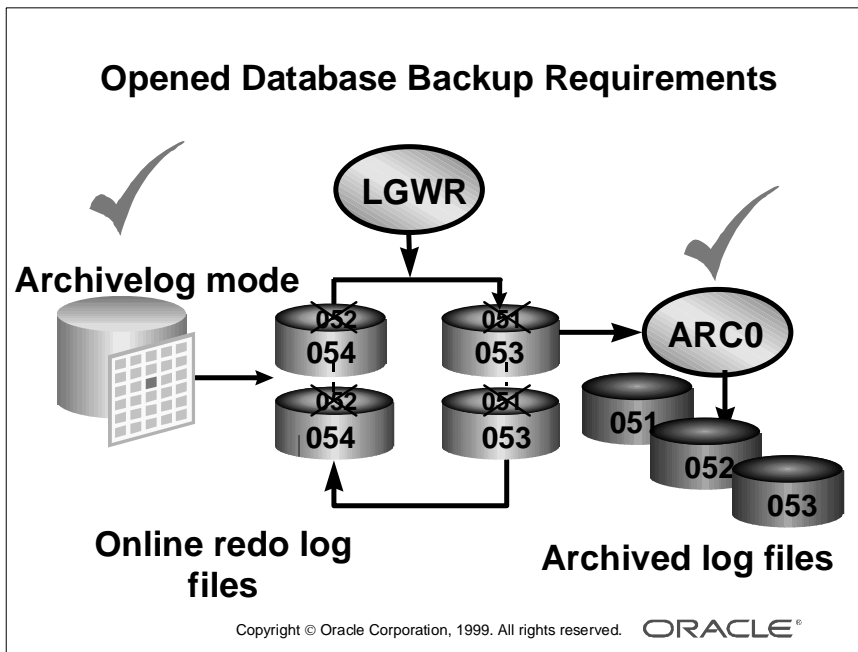
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Advantages of an Online Database Backup

- The database is available for normal use during the backup.
- A backup can be done at a tablespace or data file level.
- Supports business operations that operate all day every day.

Additional Concerns for an Online Database Backup

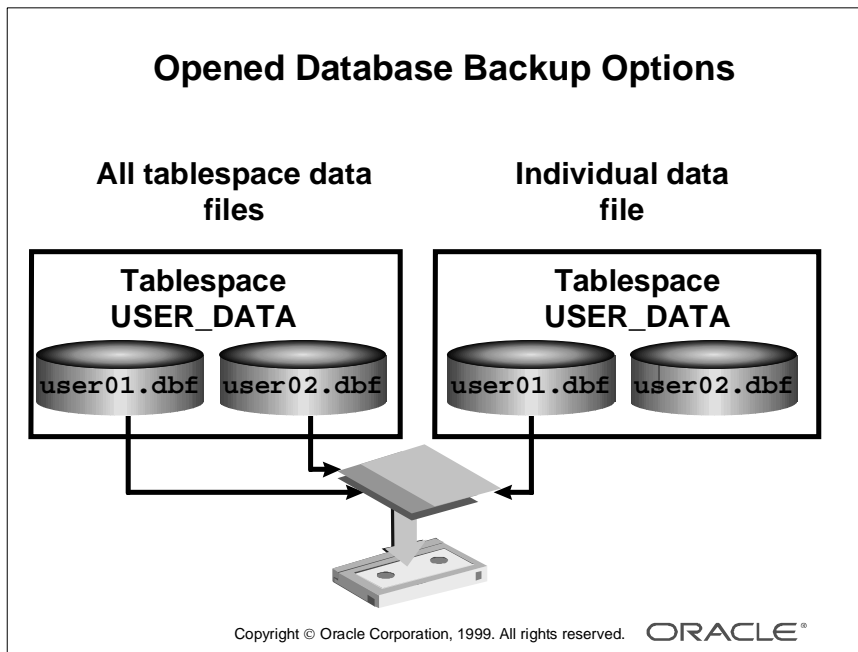
- More training is required for the DBA.
- Tested and automated scripts are recommended for performing opened database backups.



Opened Database Backup Requirements

A DBA can perform backups of tablespaces or individual data files while the database is in use, provided two criteria are met:

- Database should be set to Archivelog mode.
- It must be ensured that the Online Redo logs are archived, either by enabling the Oracle automatic archiving (ARC n) processes or by manually archiving the redo log files by using the ALTER SYSTEM ARCHIVE LOG SEQUENCE command. In an OLTP environments where redo logs are generated more frequently, it is common to enable the ARC n processes.



Opened Database Backup Options

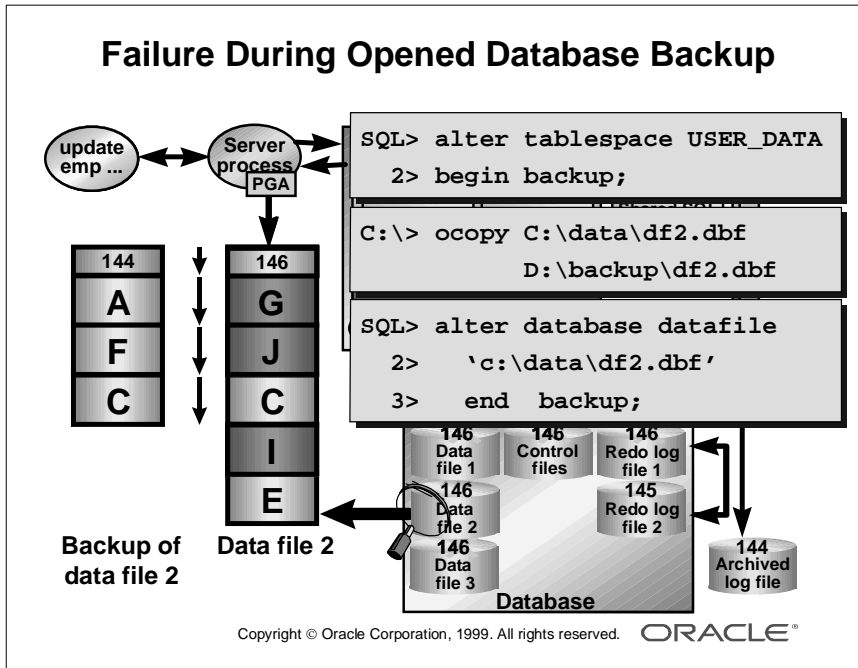
For businesses that run a 24-hour, 7-day-a-week operation, it is mandatory that backups be taken while the database is available for use. The Oracle server provides mechanisms to take a backup while the database is in full access while an opened database backup is being taken. These opened database backups taken by using the Oracle server provided mechanism are very useful for 24 x 7 type of operations.

The Oracle server enables a DBA to back up all data files for a specific tablespace, or just an individual data file for a tablespace. Regardless of the option you choose, the database remains available for normal (transaction) use during the backup process.

When a data file is placed in backup mode, more of redo log may be generated because the log writer writes block images of changed blocks of the data file in backup mode to redo log instead of just the row information.

This could have a significant impact on the size of redo logs and the performance of the log writer.

Performing an Opened Database Backup



How to Perform an Opened Database Backup

- 1 Set the data file or tablespace in backup mode by issuing the ALTER TABLESPACE...BEGIN BACKUP command. This prevents the sequence number in the data file header from changing, so that in case of recovery, logs are applied from backup start time. Even if the data file is in backup mode, it is available for normal transaction.

```
SQL> alter tablespace user_data begin backup;
```

- 2 Use an operating system backup utility to copy all data files in the tablespace to backup storage. The log sequence numbers in the backup files may be different when each tablespace is backed up sequentially.

UNIX:

```
cp /users/disk1/user01.dbf /users/backup/user01.dbf
```

NT:

```
copy c:\users\disk1\user01.ora e:\users\backup\user01.ora
```

- 3 After the data files of the tablespace have been backed up, set them into normal mode by issuing the following command:

```
SQL> alter tablespace user_data end backup;
```


How to Perform an Opened Database Backup (continued)

- 4 Force checkpoint to synchronize all the file headers through Log Switch:

```
SQL> alter system switch logfile;
```

Repeat these steps for all tablespaces, including SYSTEM, temporary tablespaces, rollback segment tablespace, and so forth.

The time between the ALTER TABLESPACE BEGIN BACKUP and ALTER TABLESPACE END BACKUP commands should be minimized, because more redo information is generated as a result of modified blocks being written to the redo log files. It is therefore recommended that you perform online backup of one tablespace at a time.

How to Use Backup Manager to Start an Online Tablespace Backup

You can also use Backup Manager to start an online tablespace backup. To perform backup of the USER_DATA tablespace follow the following steps:


- 1 Click the USER_DATA node.
- 2 Select Backup—>Begin Online Backup from the menu.
- 3 Back up all the data files belonging to the USER_DATA tablespace.
- 4 Select Backup—>End Online Backup from the menu.

Data Dictionary Views

Backup Status Information

Data dictionary views:

- V\$BACKUP
- V\$DATAFILE_HEADER



Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Dynamic Views

You can obtain information about the status of data files while performing opened database backups by querying the V\$BACKUP and V\$DATAFILE_HEADER views.

V\$BACKUP View

Query the V\$BACKUP view to determine which files are in backup mode. When an ALTER TABLESPACE BEGIN BACKUP command is issued the status changes to ACTIVE.

```
SQL> select * from v$backup;
```

FILE#	STATUS	CHANGE#	TIME
-----	-----	-----	-----
1	NOT ACTIVE	0	
2	NOT ACTIVE	0	
3	ACTIVE	240088	23/03/99

V\$BACKUP View (continued)

The Status column value will change to NOT ACTIVE once the file is backed up.

```
SQL> select * from v$backup;
```

FILE#	STATUS	CHANGE#	TIME
-----	-----	-----	-----
1	NOT ACTIVE	0	
2	NOT ACTIVE	0	
3	NOT ACTIVE	240088	23/03/99

V\$DATAFILE_HEADER View

Information about data files that are in backup mode can also be derived by querying the V\$DATAFILE_HEADER view. When an ALTER TABLESPACE BEGIN BACKUP command is issued, the value in the FUZZY column for the tablespace's data files changes to YES to indicate that the corresponding files are in backup mode.

```
SQL> select name,status,fuzzy from v$datafile_header;
```

NAME	STATUS	FUZ
-----	-----	---
/users/jdiianni/u01/sysjim01.dbf	ONLINE	
/users/jdiianni/u03/rbsjim01.dbf	ONLINE	
/users/jdiianni/u03/temp01.dbf	ONLINE	
/users/jdiianni/u03/data01_1.dbf	ONLINE	YES

The value of the FUZZY column changes to NULL when the ALTER TABLESPACE END BACKUP command is issued.

```
SQL> select name,status,fuzzy from v$datafile_header;
```

NAME	STATUS	FUZ
-----	-----	---
/users/jdiianni/u01/sysjim01.dbf	ONLINE	
/users/jdiianni/u03/rbsjim01.dbf	ONLINE	
/users/jdiianni/u03/temp01.dbf	ONLINE	
/users/jdiianni/u03/data01_1.dbf	ONLINE	YES

Backing Up a Control File

Backing Up a Control File

- Creating a binary image:

```
alter database backup controlfile to  
'control1.bkp';
```

- Creating a text trace file:

```
alter database backup controlfile to  
trace;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Backing Up a Control File

You must protect against loss of all copies of the control file. Information in the control file is required at instance startup time.

Certain status information in the control file such as the current online redo log file and the names of the database files is used by the Oracle Server during instance or media recovery. You need to maintain a recent copy of the control file after every change to the database configuration.

Guidelines

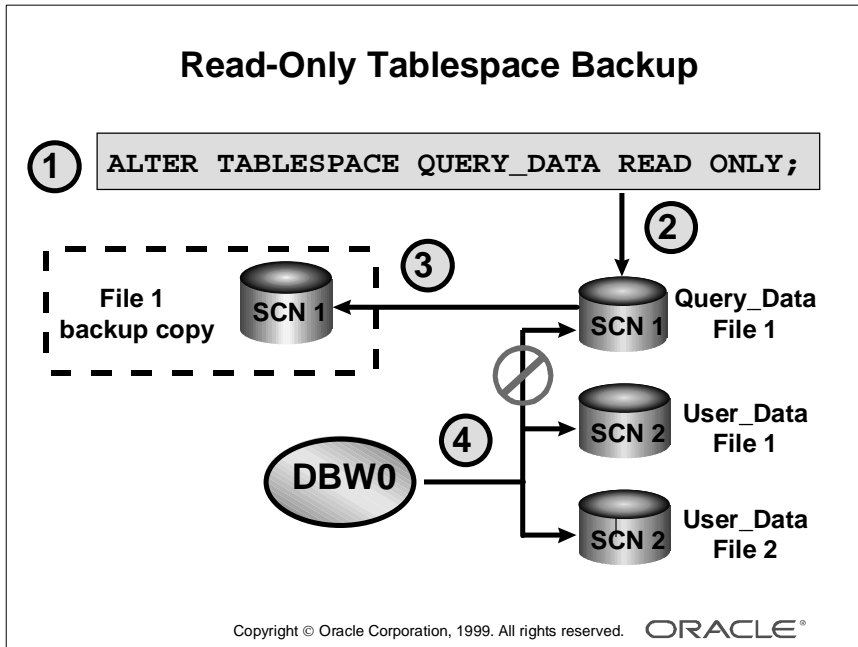
- The command ALTER DATABASE BACKUP CONTROLFILE TO TRACE provides a script to create the control file.
- In addition, the individual control files should also be backed-up by using the command ALTER DATABASE BACKUP CONTROLFILE to *<filename>*. This provides a binary copy of the control file at that time.
- Multiplex the control files and name them in the *init.ora* file by using the parameter CONTROL_FILES.
- During a full backup, shut down the instance normally and use an operating system backup utility to copy the control file to backup storage.

Commands That Change the Database Configuration

- ALTER DATABASE [ADD | DROP] LOGFILE
- ALTER DATABASE [ADD | DROP] LOGFILE MEMBER
- ALTER DATABASE [ADD | DROP] LOGFILE GROUP
- ALTER DATABASE [NOARCHIVELOG | ARCHIVELOG]
- ALTER DATABASE RENAME FILE
- CREATE TABLESPACE
- ALTER TABLESPACE [ADD | RENAME] DATAFILE
- ALTER TABLESPACE [READ WRITE | READ ONLY]
- DROP TABLESPACE

Note: It is necessary to back up the control file after one the above commands is issued.

Read-Only Tablespace Backup



Read-Only Tablespace Operations

Legend Number	Explanation
1	Change the status of a tablespace from read write to read-only by using the ALTER TABLESPACE SQL command: <code>SQL> alter tablespace query_data read only;</code>
2	When the ALTER TABLESPACE command is issued, a checkpoint is performed for all data files associated with the tablespace. The file headers are then frozen with the current SCN.
3	When you make a tablespace read-only, the DBA must perform a backup of all data files for the tablespace. An operating system copy of the files is sufficient at this point.
4	The DBW0 process writes only to data files whose tablespaces are in read write mode and normal checkpoints occur on these files.

Read-Only Tablespaces

Read-Only Tablespace Backup Issues

- Only one backup is needed after altering the tablespace to read-only
- Resume a normal backup schedule for that tablespace after making it read- write
- The control file must correctly identify read-only tablespaces, otherwise you must recover them

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Notes on Read-Only Tablespaces

- Because no writes are performed on data files for a read-only tablespace, the only time the files must be recovered is if they are damaged.
- Changing the status of a tablespace from read-only to read write results in DBW0 writing to the tablespace files and checkpoints occur as they usually would. From this point, a DBA must resume a normal backup schedule for all data files associated with the tablespace.
- The ALTER TABLESPACE command to change a tablespace to read-only updates the control file. When performing a recovery operation, the control file must correctly identify read-only tablespaces; otherwise you must recover the control file.

Logging and Nologging Options

Logging and Nologging Options	
Logging	Nologging
All changes recorded to redo	Minimal redo recorded
Fully recoverable from last backup	Not recoverable from last backup
No additional backup	May require additional backup

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Logging and Nologging Options

Tablespaces, tables, indexes, or partitions may be set to nologging mode for faster load of data by using direct-load operations. When nologging option is set for a direct-load operation, then the insert statements are not logged in the redo log files. Upon completion of the direct-load operation, the tablespace, table, index, or partition should be reset to logging mode. Because the redo logs do not contain the inserted values during the period when the table was in nologging mode, the data file pertaining to the table or partition should be backed up immediately on completion of the direct-load operation.

Note: Normal DML operations generate redo log information even if the tablespace is set to NOLOGGING.

Summary

Summary

In this lesson, you should have learned how to:

- **Understand the differences between different backup methods**
- **Know what files require backup and when to back them up**
- **Be aware of how backup methods will affect recovery operations**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Quick Reference

Context	Reference
Parameters	none
Dynamic performance views	V\$DATAFILE V\$CONTROLFILE V\$LOGFILE V\$BACKUP V\$DATAFILE_HEADER
Data dictionary views	DBA_DATA_FILES
Commands	ALTER TABLESPACE [BEGIN END] BACKUP ALTER TABLESPACE [READ WRITE READ ONLY] ALTER DATABASE BACKUP CONTROLFILE TO [options] ALTER [TABLE INDEX] [LOGGING NOLOGGING] ALTER DATABASE DATAFILE END BACKUP

Complete Recovery Without Recovery Manager

Objectives

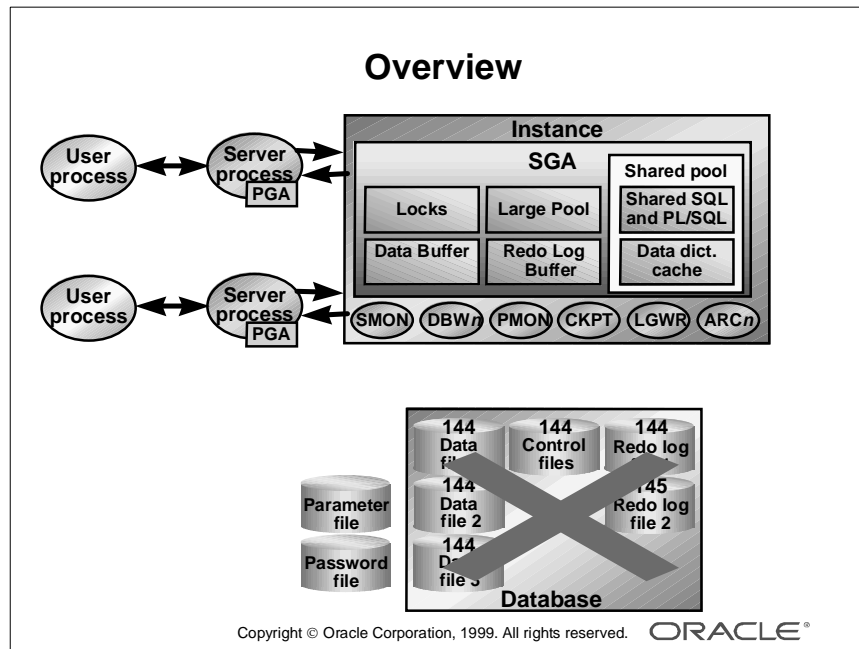
Objectives

After completing this lesson, you should be able to do the following:

- **In NOARCHIVELOG and ARCHIVELOG mode:**
 - **Note the implications of a media failure**
 - **Recover a database in different situations**
 - **Restore files to a different location if media failure occurs**
- **List the dictionary views required to recover a database after a media failure**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Overview



Overview

This lesson discusses recovery situations for databases in NOARCHIVELOG and ARCHIVELOG modes. The following factors should be considered in setting the database log mode.

- NOARCHIVELOG mode may be suitable when:
 - Data loss between backups can be tolerated (development, training, etc.,)
 - Recovery is faster by reapplying transactions (from batch files)
 - Data rarely changes (non-OLTP)
- ARCHIVELOG mode is preferable when:
 - Database cannot be shut down for closed backup
 - Data loss cannot be tolerated
 - It is easier to recover by using archivelogs than applying transactions (OLTP)
- By default, the database is in NOARCHIVELOG mode.

Media Failure

Media Failure and Recovery: Database in NOARCHIVELOG Mode

- **Failure:** loss of disk, data file, or corruption
- **Recovery:**
Restore all Oracle files:
 - Data files
 - Control files
 - Redo log files
 - Password file (optional)
 - Parameter file (optional)

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Media Failure

When media failure occurs a valid closed database backup must exist in order to recover, since all Oracle files must be restored, even if only one data file is damaged or lost. Make sure the following files are restored:

- All data files, control files, and redo logs. Remember, all Oracle files must be synchronized for the database to open.
- Password or parameter files only if they are corrupt or lost.

Note: For a database in NOARCHIVELOG mode, you do not have to restore all Oracle files if no redo log file has been overwritten since the last backup.

- Scenario
 - There are two redo logs for a database.
 - A closed database backup was taken at log sequence 144.
 - While the database was at log sequence 145, data file 2 was lost.

- Result

Since log sequence 144 has not been overwritten, just data file number 2 can be restored and recovered manually.

Recovery: NOARCHIVELOG Mode

Advantages

- **Easy to perform, with low risk of error.**
- **Recovery time is the time it takes to restore all files.**

Disadvantages

- **Data is lost and must be reapplied manually.**
- **The entire database is restored to the point of the last whole closed backup.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Advantages and Disadvantages

If you decide to operate databases such as test or development in NOARCHIVELOG mode, consider the following advantages and disadvantages:

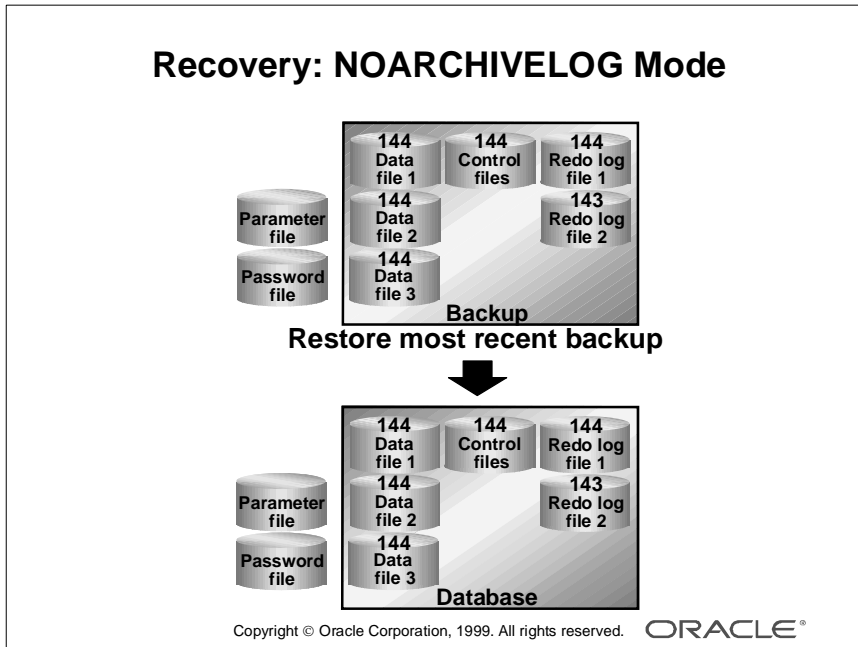
Advantages

- Easy to perform, since only a restore of all files from a backup is required. The only risks are restoring the wrong backup, overwriting the backup, not shutting down the database before restore, or invalid backups, that training and procedures can easily solve.
- The major component of recovery time is merely the length of time your hardware can restore all files.

Disadvantages

- All data entered by users since the last backup will be lost and must be reapplied manually.
- The entire database has to be restored from the last whole closed backup, even if only one data file is lost.

Recovery: NOARCHIVELOG Mode



Recover from a Media Failure (NOARCHIVELOG Mode)

- 1 Disk 2 is damaged, losing data file number 2. Only two log files exist.
- 2 Since the last backup was taken at log sequence 144 and the current log sequence number is 146, we cannot recover the data file, since redo log 144 is overwritten (this would be confirmed if recovery was attempted). Therefore, shut down the database and restore all Oracle files.

```
SQL> shutdown abort;
```

To restore files:

```
UNIX > host cp /disk1/backup/* /disk1/data/
```

```
NT > host copy \disk1\backup\*. * \disk1\data\
```

- 3 When the copy is finished, restart the instance:

```
SQL> connect / as sysdba;
```

```
SQL> startup pfile=initDB00.ora;
```

- 4 Notify users that they will need to reenter data from the time of the last backup.

Restoring Data Files

Restoring to a Different Location

Rename the file or directory location

```
SQL> connect system/manager as sysdba;
Connected.
SQL> startup mount pfile=initDB00.ora;
Oracle instance started.
SQL> alter database rename file
      2>    '/disk1/data/user_01.dbf'
      3> to '/disk2/data/user_01.dbf';
Statement Processed.
SQL> alter database open;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Restore Files to a Different Location

- 1 If the control files are restored to a different location, update the parameter file.
- 2 If a data file or redo log is restored to a different location or name, then:
 - Mount the instance.
 - Use the ALTER DATABASE command to update the control file with the new file location:

```
SQL> alter database rename file
      2  '/disk1/data/user_01.dbf'
      3 to '/disk2/data/user_01.dbf';
```

Note: In the UNIX environment, the files must exist in the new location prior to issuing the ALTER DATABASE RENAME command. This is not the case in an NT environment.

Complete Recovery

Media Failure and Recovery: ARCHIVELOG Mode

- **Failure: loss of disk, data file, or corruption**
- **Recovery**
 - **Data files for restore must be offline.**
 - **Restore only lost or damaged data files.**
 - **Do not restore the control files, redo log files, password files, or parameter files.**
 - **Recover the data files.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Complete Recovery

When media failure occurs with a database in ARCHIVELOG mode, in order to recover completely up to the time of failure, you must have the following:

- A valid backup containing the lost or damaged data files after database was set in ARCHIVELOG mode.
- All archived logs from the backup you are restoring to the present time.
- The redo log files that contain the transactions that are not archived yet.

How to Recover from a Media Failure

If you meet the above requirements for complete recovery, then follow these steps to recover data files:

- 1** Make sure that files to be overwritten are not opened during restore. Query the V\$DATAFILE and V\$TABLESPACE views to ascertain the status of the file.
- 2** Make sure you only restore from backup the file that is now lost or damaged. Remember, restoring all files will take your database back in time. Make sure that you do not restore the online redo log files.
- 3** Place the database in either mount or open mode.
- 4** Recover the data files by using the recover command.

Recovery in ARCHIVELOG Mode (Complete Recovery)

Advantages

- Only need to restore lost files
- Recovers all data to the time of failure
- Recovery time is the time it takes to restore lost files and apply all archived log files

Disadvantage

Must have all archived log files since the backup from which you are restoring

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Advantages and Disadvantages

The following are advantages and disadvantages of running your database in ARCHIVELOG mode.

Advantages

- Only need to restore lost or damaged files.
- No committed data is lost. Restoring the files, then applying archived and redo logs, brings the database to the current point-in-time.
- The total recovery time is the length of time your hardware can restore the required files and apply all archived and redo logs.
- Recovery can be performed while the database is open (except system tablespace files and data files containing online rollback segments).

Disadvantage You must have all archived logs from the time of your last backup to the current time. If you are missing one, you cannot perform a complete recovery, since all archives need to be applied in sequence; that is, archived log 144, then 145, then 146, and so on.

Complete Recovery Methods

- **Closed database recovery for:**
 - System data files
 - Rollback segment data files
 - Whole database
- **Opened database recovery, with database initially opened: for file loss**
- **Opened database recovery with database initially closed: for hardware failure**
- **Data file recovery with no data file backup**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Complete Recovery Methods

There are four methods for performing complete recovery:

Method 1: Recovering a Closed Database This method of recovery generally uses either the RECOVER DATABASE or RECOVER DATAFILE commands when:

- The database is not operational a 24 hour a day, 7 days a week.
- The recovered files belong to the system or rollback segment tablespace.
- The whole database, or a majority of the data files, need recovery.

Method 2: Recovering an Opened Database, Initially Opened This method of recovery is generally used when:

- File corruption, accidental loss of file, or media failure has occurred, which has not resulted in the database being shut down.
- The database is operational a 24 hour a day, 7 days a week. Downtime for the database must be kept to a minimum.
- Recovered files do not belong to the system or rollback tablespaces.

Complete Recovery Methods (continued)

Method 3: Recovering an Opened Database, Initially Closed This method of recovery is generally used when:

- A media or hardware failure has brought the system down.
- The database is operational a 24 hour a day, 7 days a week database. Down-time for the database must be kept to a minimum.
- The restored files do not belong to the system or rollback tablespace.

Method 4: Recovering a Data File with No Backup This method of recovery is generally used when:

- Media or user failure has resulted in loss of a data file that was never backed up.
- All archived logs exist since the file was created.
- The restored files do not belong to the system or rollback tablespace.

Note: During recovery, all archived logs files need to be available to the Oracle server on disk. If they are on a backup tape, you must restore them first.

Recover Syntax

Recover a mounted database:

```
SQL> recover database;  
SQL> recover datafile '/disk1/data/df2.dbf';  
SQL> alter database recover database;
```

Recover an opened database:

```
SQL> recover tablespace USER_DATA;  
SQL> recover datafile 2;  
SQL> alter database recover datafile 2;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Recover Syntax

One of the following commands may be issued to recover the database:

- RECOVER [*AUTOMATIC*] DATABASE
Can only be used for a closed database recovery.
 - RECOVER [*AUTOMATIC*] TABLESPACE <NUMBER> | <NAME>
Can only be used for an opened database recovery.
 - RECOVER [*AUTOMATIC*] DATAFILE <NUMBER> | <NAME>
Can only be used for both an opened and closed database recovery.
- where: automatic automatically applies archived and
redo log files.

Note: ALTER DATABASE may be placed in front of the RECOVER command. This is not recommended because some error messages get suppressed and do not show up on screen during recovery.

Recovery by Using Archived Log Files

Recovery Using Archived Logs

- To change archive location, use the **ALTER SYSTEM ARCHIVE LOG . . .** command.
- To automatically apply redo log files:
 - Issue **SET AUTORECOVERY ON** before starting media recovery (**RECOVER**).
 - Enter **auto** when prompted for an archived log file.
 - Use the **RECOVER AUTOMATIC . . .** command.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Recovery by Using Archived Log Files

During recovery, the Oracle server can manually or automatically apply the necessary archived and redo log files to reconstruct the data files. Before a redo log file is applied, the Oracle server suggests the log file name to apply.

Restoring Archives to a Different Location

If archived logs are not restored to the `LOG_ARCHIVE_DEST` directory, then the Oracle server will need to be notified before or during recovery, by:

- Specifying the location and name at the recover prompt:
`Specify log: {<RET>=suggested | filename | AUTO | CANCEL}`
- Use the **ALTER SYSTEM ARCHIVE** command:
`SQL> alter system archive log start to <new location>;`
- Use the **RECOVER FROM <LOCATION>** command:
`SQL> recover from '<new location>' database;`

How to Apply Redo Log Files Automatically

- 1 Before starting media recovery, issue the SQL*Plus statement:

```
SQL> set autorecovery on
```

- 2 Enter auto when prompted for a redo log file:

```
SQL> recover datafile 4;
```

```
ORA-00279: change 308810...12/02/97 17:00:14 needed for thread 1
```

```
ORA-00289: suggestion : /disk1/archive/arch_35.rdo
```

```
ORA-00280: change 308810 for thread 1 is in sequence #35
```

```
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

```
AUTO
```

```
Log applied.
```

```
...
```

- 3 Use the AUTOMATIC option of the recovery command:

```
SQL> recover automatic datafile 4;
```

```
Media recovery complete.
```


Locating Data Files

Files Needed for Recovery

- View `V$RECOVER_FILE` to locate data files needing recovery.
- View `V$LOG_HISTORY` for a list of all archived logs for the database.
- View `V$RECOVERY_LOG` for a list of all archived logs required for recovery.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Locating Data Files That Need Recovery

To locate data files needing recovery, and where they need recovery from, use the `V$RECOVER_FILE` view.

```
SQL> select * from v$recover_file;
```

FILE#	ONLINE	ERROR	CHANGE#	TIME
-----	-----	-----	-----	-----
2	OFFLINE		288772	02-MAR-99

- The `ERROR` column returns two possible values to define the reason why the file needs to be recovered:
 - `NULL` if the reason is unknown
 - `OFFLINE NORMAL` if recovery is not needed
- The `CHANGE#` column returns the SCN (system change number) where recovery must start.

Locating Archived Log Files to Apply

To locate archived log files, view V\$ARCHIVED_LOG for all archives or V\$RECOVERY_LOG for archives needed during recovery:

```
SQL> select * from v$recovery_log;
```

THREAD#	SEQUENCE#	TIME	ARCHIVE_NAME
1	34	02-MAR-99	/disk1/archive/arch_34.rdo
...			
1	43	04-MAR-99	/disk1/archive/arch_43.rdo
1	44	04-MAR-99	/disk1/archive/arch_44.rdo

```
SQL> recover datafile 2;
```

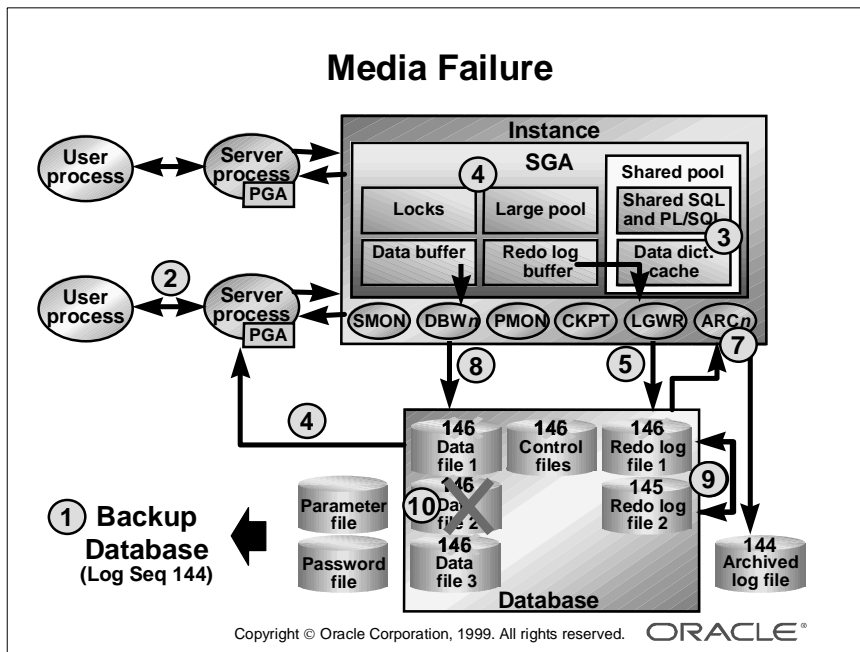
```
ORA-00279: change 288772...03/02/99 15:32:29 needed for thread 1
```

```
ORA-00289: suggestion : /disk1/archive/arch_34.rdo
```

```
ORA-00280: change 288772 for thread 1 is in sequence #34
```

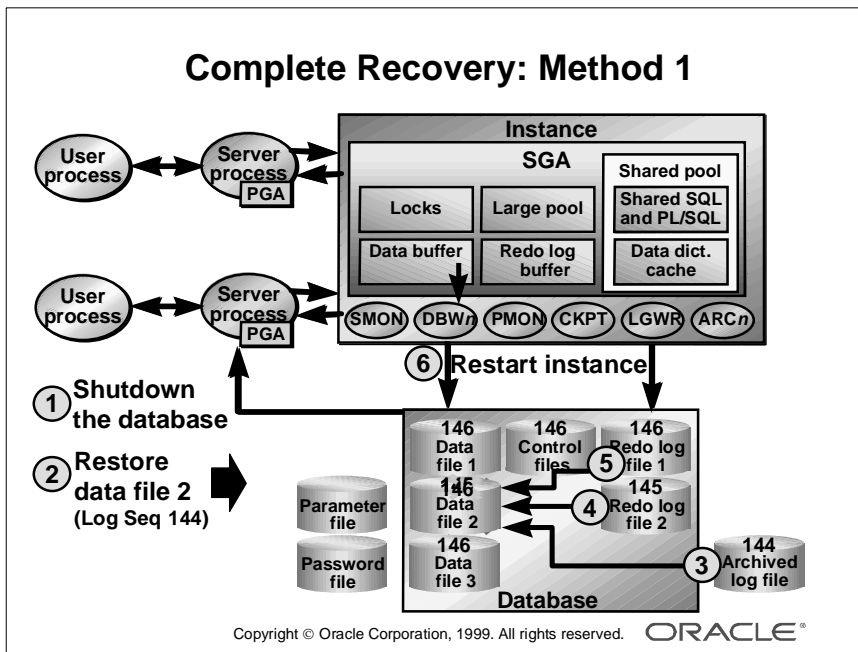
```
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

From the above information, archived logs from 34 on are required to completely recover data file 2.



Recovery Methods

Each of the four recovery methods are discussed below.



Complete Recovery: Method 1 (Mounted Database)

Further investigation reveals that corrupt blocks were found on disk 2 where data file 2 is stored. Viewing V\$DATAFILE and V\$TABLESPACE views, you have determined that data file 2 is one of the files belonging to the system tablespace. Therefore method 1 must be used:

- 1 Restore the file from backup (the most recent if available):

```
UNIX> host cp /disk1/backup/df2.dbf /disk2/data/
NT > host copy c:\backup\df2.dbf d:\data\
```

- 2 Start the instance in mount mode and recover the data file:

```
SQL> startup mount pfile=initDB00.ora
or SQL> recover datafile '/disk2/data/df2.dbf';
ORA-00279: change 148448 ...11/29/97 17:04:20 needed for thread
ORA-00289: suggestion : /disk1/archive/arch_6.rdo
ORA-00280: change 148448 for thread 1 is in sequence #6
Log applied.
...
Media recovery complete.
```

- 3 To bring the data file to the point of failure, all needed archived logs and redo logs are applied.

Complete Recovery: Method 1 (Mounted Database) (continued)

- 4 When recovery is finished, all data files are synchronized. Open the database.

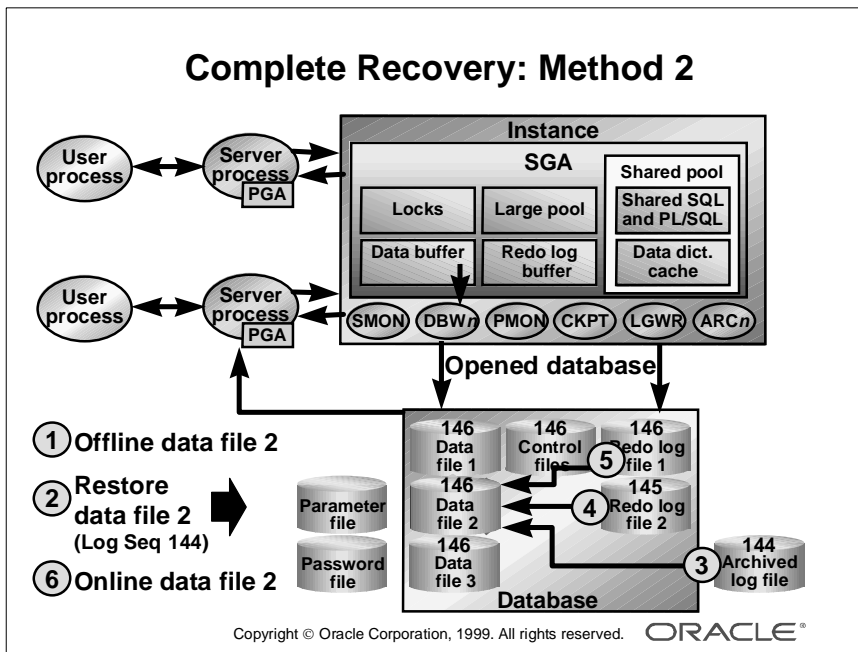
```
SQL> alter database open;
```

Afterwards, notify users that the database is available for use, and tell them to reenter any data that was not committed before system failure.

Note

- If there are many archived logs to apply, use either the AUTOMATIC feature of the RECOVER command or use the following SQL*Plus command:

```
SQL> set autorecovery on  
SQL> recover database;
```
- During this method of recovery, the database must be closed; the entire database is inaccessible to users during the recovery process.
- If there is enough space available, restore the required archived redo log files to the location currently specified by LOG_ARCHIVE_DEST.



Complete Recovery: Method 2 (Opened Database, Initially Opened)

Your training DBA immediately informs you that it was not media failure—he accidentally removed data file number 2 by using operating system commands. The database is currently opened, so to determine which tablespace the data file belongs to, use the following command:

```
SQL> select file_id f#, file_name,
2> tablespace_name tablespace, status
3> from dba_data_files;
```

F#	FILE_NAME	TABLESPACE	STATUS
1	/disk1/data/system_01.dbf	SYSTEM	AVAILABLE
2	/disk2/data/df2.dbf	USER_DATA	AVAILABLE
3	/disk1/data/rbs01.dbf	RBS	AVAILABLE
...			

Complete Recovery: Method 2 (Opened Database, Initially Opened) (continued)

- 1 Since the data file is not a system or rollback segment data file, we can use Method 2. Let's determine whether we need to take data file 2 offline (in this case, the Oracle server has already taken the file offline):

```
SQL> select d.file# f#, d.name, d.status, h.status
       2 from v$datafile d, v$datafile_header h
       3 where d.file# = h.file#;
```

F#	D.NAME	D.STATUS	H.STATUS
1	/disk1/data/system_01.dbf	SYSTEM	ONLINE
2	/disk2/data/df2.dbf	RECOVER	OFFLINE
3	/disk1/data/rbs_01.dbf	ONLINE	ONLINE

...

- 2 Since the file is offline, the file can now be restored successfully:

```
for UNIX > host cp /disk1/backup/df2.dbf /disk2/data/
for NT      > host copy c:\backup\df2.dbf d:\data\
```

- 3 Use the RECOVER or ALTER DATABASE RECOVER commands to apply the archives and the redo logs to the restored data file.

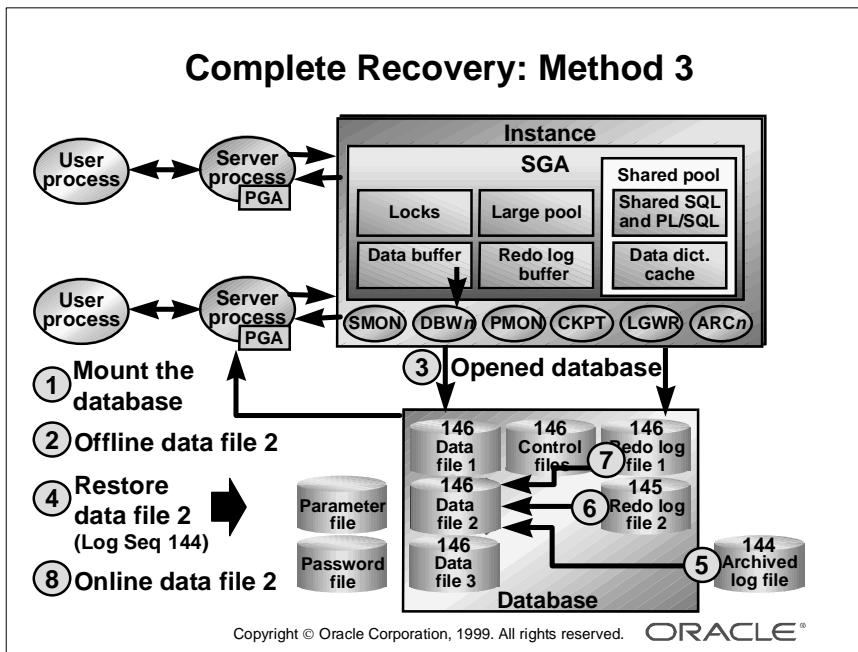
```
SQL> recover datafile '/disk2/backup/df2.dbf';
or SQL> recover tablespace USER_DATA;
```

- 4 When recovery is finished, all data files are synchronized. Bring the data file online:

```
SQL> alter database datafile '/disk2/data/df2.dbf' online;
or SQL> alter tablespace USER_DATA online;
```

Note

- Oracle will sometimes detect a file problem and automatically take it offline. Before recovery, always check the alert log for any errors and check the status of files—OFFLINE files may require recovery.
- When a tablespace is taken offline, all data files are taken offline and no data contained in that tablespace can be accessed. For a multifile tablespace, when one data file is taken offline, only data contained inside that data file cannot be accessed. The tablespace still remains available.



Complete Recovery: Method 3 (Opened Database, Initially Closed)

From your investigation, you have just determined that the media failure was due to a failed disk controller, which luckily contains only disk 2. From your familiarity with the database, you know data file 2 is not a system or rollback segment data file, nor will it prevent users running their end-of-month reports.

- 1 Mount the database. It will not open because data file 2 cannot be opened.

```
SQL> startup mount pfile=$HOME/initDB00.ora
```

Database mounted.

If you are not sure of the tablespace number to which the file belongs:

```
SQL> select d.file#, d.ts#, h.tablespace_name, d.name,
2      h.error
3      from v$datafile d, v$datafile_header h
4      where d.file# = h.file#;
```


Complete Recovery: Method 3 (Opened Database, Initially Closed) (continued)

FILE#	TS#	TABLESPACE	NAME	Error
1	0	SYSTEM	/disk1/data/system01.dbf	
2	1		/disk2/data/df2.dbf	FILE NOT FOUND
3	2	RBS	/disk1/data/rbs01.dbf	
...				

- 2** If the data file is not offline, the database will not open. Therefore, the file must be taken offline. You have queried V\$DATAFILE and determined that the file is online. The following command must be issued:

```
SQL> alter database datafile '/disk2/data/df2.dbf' offline;
```

Note: The ALTER TABLESPACE command cannot be used here since the database is not opened yet.

- 3** The database can now be opened:

```
SQL> alter database open;
```

- 4** Now that users can access the system, restore the file. Since it cannot be restored to the damaged disk 2, restore it to disk 3:

```
for UNIX > host cp /disk1/backup/df2.dbf /disk3/data/
```

```
for NT> host copy c:\backup\df2.dbf e:\data\
```

The Oracle server must now be informed of the new file location:

```
SQL> alter database rename file '/disk2/data/df2.dbf'
2 to '/disk3/data/df2.dbf';
```

When the database is opened and tablespace recovery is required, to determine the name of the tablespace that owns the data file:

```
SQL> select file_id f#, file_name,
2 tablespace_name tablespace, status
3 from dba_data_files;
```

F#	FILE_NAME	TABLESPACE	STATUS
1	/disk1/data/system_01.dbf	SYSTEM	AVAILABLE
2	/disk3/data/df2.dbf	USER_DATA	AVAILABLE
3	/disk1/data/rbs01.dbf	RBS	AVAILABLE

Complete Recovery: Method 3 (Opened Database, Initially Closed) (continued)

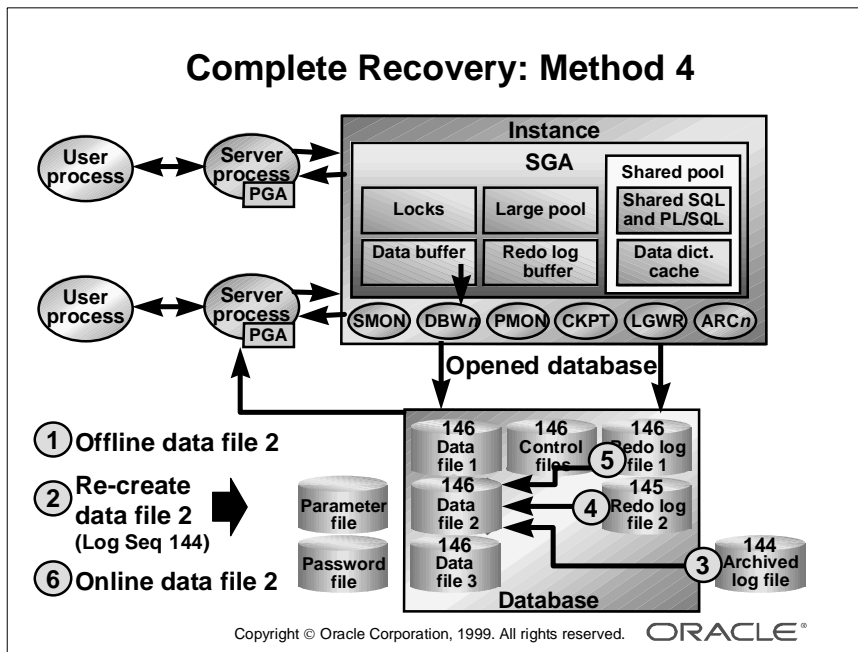
- 5** Use the RECOVER or ALTER DATABASE RECOVER commands to start applying the archives and the redo logs to the restored data file.

```
SQL> recover datafile '/disk3/data/df2.dbf';  
or SQL> recover tablespace USER_DATA;
```

- 6** When recovery is finished, all data files are synchronized. Bring the data file online:

```
SQL> alter database datafile '/disk3/data/df2.dbf' online;  
or SQL> alter tablespace USER_DATA online;
```

- 7** Notify users that the database is available for use, and tell them to reenter any data that was not committed before system failure.



Complete Recovery: Method 4 (Loss of Data File with No Backup)

Because data file 7 (in disk 1) is lost, you immediately go to your backup tape. However, during restore, you receive an error indicating that the file was not backed up. You locate the DBA who created the TABLE_DATA tablespace two days ago and found that it contained important user data, that was never included in the backup strategy. Since data file 7 is not a system or rollback segment data file, and we have all archived logs for the past two days, Method 4 is the best choice:

- 1 If the database is closed, then mount the database, take the data file (with no backup) offline, and open the database. This allows users, who do not need the TABLE_DATA tablespace, to work on the system. If the database is opened, offline the data file. For variety, set the tablespace offline, because it only contained one data file.

Note: The immediate option must be included, if the database is opened, to avoid a checkpoint trying to write to a file that does not exist:

```
SQL> alter tablespace TABLE_DATA offline immediate;
Tablespace altered.
```

**Complete Recovery: Method 4 (Loss of Data File with No Backup)
(continued)**

You confirm the recovery status by querying V\$RECOVER_FILE to check the status of a backup:

```
SQL> select * from v$recover_file;
```

FILE#	ONLINE	ERROR	CHANGE#	TIME
7	OFFLINE	FILE NOT FOUND	0	

2 You now need to re-create the file:

```
SQL> alter database create datafile '/disk2/DATA/df7.dbf'
      2 as '/disk1/DATA/df7.dbf';
```

Database altered.

```
SQL> select * from v$recover_file;
```

FILE#	ONLINE	ERROR	CHANGE#	TIME
7	OFFLINE		248621	01-DEC-97

3 Use the RECOVER or ALTER DATABASE RECOVER commands to start applying the archives and the redo logs to the recreated data file:

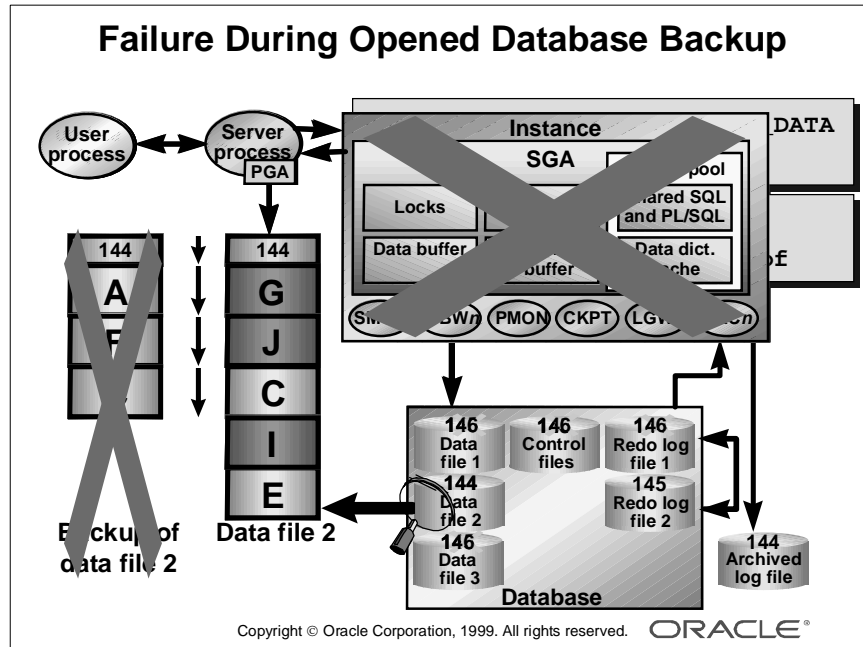
```
SQL> recover tablespace TABLE_DATA;
```

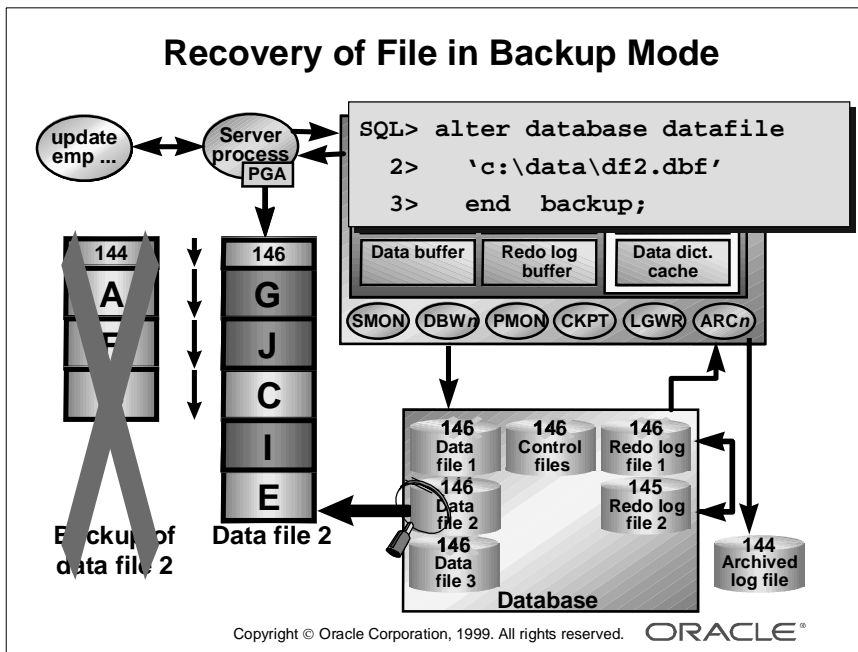
4 To bring the data file to the point of failure, all needed archived logs and redo logs are applied.**5** All data files are now synchronized.**6** When recovery is finished, bring the data file online (we could bring just one data file online; remember that we are using the tablespace in step 1 instead of data file):

```
SQL> alter tablespace TABLE_DATA online;
```

All data is now recovered. Include the file in the backup strategy and notify users that the tablespace is ready to be used again.

Recovery After Failure of Opened Database Backup





Failure During an Opened Database Backup

During an opened database backup, the system might crash, a power failure may occur, the database is shut down, and so on. If any of these occurs:

- The backup files will be unusable if the operating system did not complete the backup. You will need to backup the files again.
- The database files in hot backup mode will not be synchronized with the database, since the header is frozen when the backup starts.

Recovery of Data File in Backup Mode

There are two methods for recovering from this scenario:

- Recover the database files by using a method already discussed.
- Find another way to synchronize the files.

The Problem

The ALTER TABLESPACE command cannot be issued until the database is opened, and the database cannot be opened until files are synchronized or offline. Taking the files offline is no use, since an ALTER TABLESPACE... END BACKUP cannot be performed on data files that are offline.

The Solution

If you are unsure whether a file needs to be recovered, or it was left in hot backup mode, query the V\$BACKUP view:

```
SQL> select * from v$backup;
```

FILE#	STATUS	CHANGE#	TIME
1	NOT ACTIVE	0	
2	ACTIVE	228596	30-NOV-97
3	NOT ACTIVE	0	
4	NOT ACTIVE	0	

This output indicates that file number 2 is currently in hot backup mode. To unfreeze the header, issue the command:

```
SQL> alter database datafile 2 end backup;
Database altered.
```

```
SQL> select * from v$backup;
```

FILE#	STATUS	CHANGE#	TIME
1	NOT ACTIVE	0	
2	NOT ACTIVE	228596	30-NOV-97
...			

All that needs to be done now is to open the database for users:

```
SQL> alter database open;
```

Clearing Corrupted Redo Logs

Clearing Redo Log Files

Using the unarchived option:

```
SQL > alter database clear unarchived  
2 logfile group 1;
```

Using the unrecoverable datafile option:

```
SQL > alter database clear unarchived  
2 logfile group 1  
3 unrecoverable datafile;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Clearing Corrupted Redo Logs

If an online redo log file has been corrupted while the database is opened, ALTER DATABASE CLEAR LOGFILE can be used to create or clear the files without the database needing to be shut down.

Find the status of the redo log files:

```
SQL> select * from v$logfile;  
  
GROUP#    STATUS    MEMBER  
-----  
2         STALE    /disk1/DATA/log2a.rdo  
1         STALE    /disk1/DATA/log1a.rdo
```

The status of a log member can be:

- **INVALID:** The file is inaccessible.
- **STALE:** The file's contents are incomplete because it is not filled up yet.
- **Blank:** The file is in use.

ALTER DATABASE CLEAR UNARCHIVED LOGFILE Command

This command overcomes two situations where dropping redo logs is not possible:

- If there are only two logs groups.
- The corrupt redo log file belongs to the current group.

It also provides an efficient method for recreating damaged log files and clearing corrupted log files while the database is opened.

Note

- Use this command cautiously. If no archived log was produced, complete recovery is not possible. Perform a backup after completion of the command.
- A logfile currently required for recovery cannot be cleared.

Clearing Logs Required by Offline Data Files

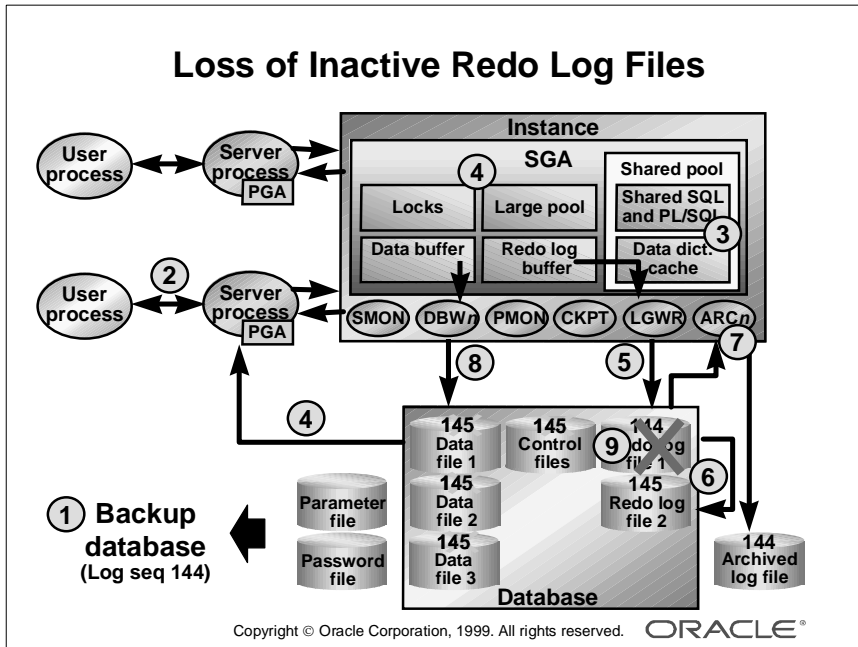
Use the ALTER DATABASE CLEAR LOGFILE ... UNRECOVERABLE DATAFILE command to clear a redo log file, even if offline data files require the log for recovery. This situation is rare, though if it occurs there are two approaches:

- Restore all data files and perform an incomplete recovery prior to the cleared log file.
- or*
- Drop the tablespace containing the unrecoverable data files.

Note

- Offline data files requiring this log will be unusable after the command.
- An archived log will probably not exist for the cleared log file.
- Consider taking a backup immediately after issuing this command.

Loss of Inactive Redo Log File



Loss of Inactive Redo Logs

If redo logs are lost or corrupted, recovery to the time of failure may not be possible. However, no data is lost in the following circumstances:

- The redo logs files lost are not current.
- The redo log has been archived.
- The database is properly configured with mirrored redo log files.

Determining Need for Recovery

You have just spent four hours creating a database and decide to put the database in ARCHIVELOG mode. Being a conscientious and thorough DBA, you decide to test the creation of archived logs on the system. You use the ALTER SYSTEM command to manually switch the logs. After the second time, you receive the following error message:

```
SQL> alter system switch logfile;
ORA-00470: LGWR process terminated with error
```

Determining Need for Recovery (continued)

You immediately realize that a background process abnormally terminated and aborts the instance. You have no backup, and you do not want to spend another four hours recreating the database. So, you restart the instance:

```
SQL> startup pfile=$HOME/initDB00.ora
ORACLE instance started.

...
ORA-00313: open failed for members of log group 1 of thread 1
ORA-00312: online log 1 thread 1: '/disk2/DATA/log1a.rdo'
```

To determine the severity of your redo log situation, you decide to query V\$LOG and make three important observations:

```
SQL> select * from v$log;
GROUP#  THREAD#  ...BYTES  MEMBERS    ARC    STATUS  FIRST_CHAN...
-----  -
1         1    153600      1    YES    UNUSED          0
2         1    153600      1    NO     CURRENT    248720
```

- The FIRST_CHANGE number is 0, indicating a problem with log group 1.
- Log group 2 is the current group; therefore, group 1 is not active.
- The file for group 1 has been archived (ARC column = YES). Therefore, we have not lost any recovery information.

You try to locate the operating system file, but it is not there. Because no information is lost (only a log file), no recovery is required—only the log file must be re-created.

Re-creating Redo Logs

Inactive Redo Log Recovery

Re-creating log files:

- Find a location to create the redo log file.
- Drop the log group containing the lost file.
- Re-create the log group.
- Open the database.

Alternatively, to open the database faster, use the clear logfile command.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Re-creating Redo Logs

The database cannot be opened until the redo logs have been re-created. This can quickly be done by using the following commands:

- 1 Find the location of where the file was previously located:

```
SQL> select * from v$logfile;
  GROUP#      STATUS      MEMBER
  -----      -
          2      STALE      /disk1/DATA/log2a.rdo
          1
          1      /disk1/DATA/log1a.rdo
```

- 2 You cannot drop a log group, since there must always be at least two log groups:

```
SQL> alter database drop logfile group 1;
alter database drop logfile group 1
*
ORA-01567: dropping log 1 would leave less than 2 log files ...
ORA-00312: online log 1 thread 1: '/disk1/DATA/log1a.rdo'
```

Therefore, we must create another temporary group:

```
SQL> alter database add logfile group 3
      > '/disk1/DATA/log3a.rdo' size 150k;
Database altered.
```

Re-creating Redo Logs (continued)

- 3** Now drop the log group:

```
SQL> alter database drop logfile group 1;  
Database altered.
```

- 4** Open the database:

```
SQL> alter database open;
```

- 5** Immediately multiplex all redo logs. This will reduce the chance of losing data.

Clearing Online Redo Logs

When both log groups are the same size, steps 2 through 7 can be combined into two simple commands:

```
SQL> host cp /disk1/DATA/log2a.rdo /disk1/DATA/log1a.rdo  
SQL> alter database clear logfile '/disk1/DATA/log1a.rdo';
```

Recovery Status Information

Viewing Recovery Information

V\$RECOVERY_FILE_STATUS

- Files needing recovery
- Status of recovery

V\$RECOVERY_STATUS

- Start time of recovery
- Log sequence number needed
- Status of previous log applied
- Reason recovery needs user input

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Recovery Status Information

There are two database views that provide status information to the server process and user performing media recovery:

- **V\$RECOVERY_STATUS**: Contains overall database recovery information.
- **V\$RECOVERY_FILE_STATUS**: Contains information for each data file requiring recovery.

Note: The information for these views resides in the PGA of the server process when the ALTER DATABASE RECOVER command is issued. No other sessions can therefore see recovery information.

Recovery Status Example

From the previous example where we lost data file 2, we will now issue the recovery commands while viewing the recovery information:

```
SQL> alter database recover datafile 2;
alter database recover datafile 2
*
ORA-00279: change 148448...11/29/97 17:04:20 needed for thread 1
ORA-00289: suggestion : /disk1/archive/arch_6.rdo
ORA-00280: change 148448 for thread 1 is in sequence #6
```

```
SQL> select * from v$recovery_status;
RECOVERY  THREAD  SEQ_NEEDED  SCN_NEEDED  TIME_NEED  PREV_LOG
-----  -
29-NOV-99      1           6           0    29-NOV-97    NONE
1 row selected.
```

```
SQL> select * from v$recovery_file_status;
FILENUM  FILENAME                STATUS
-----  -
        6  /disk1/data/df2.dbf    IN RECOVERY
1 row selected.
```

Perform the actual recovery:

```
SQL> alter database recover
      2      automatic logfile 'disk1/archive/arch_6.rdo';
ORA-00279: change 148448 ...11/29/97 17:04:20 needed for thread
ORA-00289: suggestion : /disk1/archive/arch_6.rdo
ORA-00280: change 148448 for thread 1 is in sequence #6
Log applied.
...
Media recovery complete.
SQL> select * from v$recovery_status;
RECOVERY  THREAD  SEQ_NEEDED  SCN_NEEDED  TIME_NEED  PREV_LOG
-----  -
0 rows selected.
SQL> select * from v$recovery_file_status;
FILENUM  FILENAME                STATUS
-----  -
0 rows selected.
```

Summary

Summary

In this lesson, you should have learned how to:

- **Recover a database in NOARCHIVELOG mode: restoring all Oracle files from backup.**
- **Restore files to new locations.**
- **Perform complete recovery which involves recovering to the time of failure.**
- **Determine suitability of database in ARCHIVELOG mode for complete recovery.**
- **Perform recovery by using open database backup.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Quick Reference

Context	Reference
Parameters	none
Dynamic performance views	V\$RECOVER_FILE V\$LOG_HISTORY V\$RECOVERY_LOG V\$RECOVERY_FILE_STATUS V\$RECOVER_STATUS V\$BACKUP V\$LOGFILE V\$LOG
Data dictionary views	none
Commands	ALTER DATABASE DATAFILE ONLINE ALTER DATABASE DATAFILE OFFLINE ALTER DATABASE RENAME FILE ALTER DATABASE OPEN ALTER DATABASE RECOVER DATABASE ALTER DATABASE RECOVER DATAFILE ALTER SYSTEM ARCHIVE LOG ALTER TABLESPACE BEGIN BACKUP ALTER TABLESPACE END BACKUP ALTER DATABASE DATAFILE END BACKUP ALTER DATABASE ADD LOGFILE GROUP ALTER DATABASE DROP LOGFILE GROUP ALTER DATABASE CLEAR LOGFILE ALTER DATABASE CLEAR UNARCHIVED LOGFILE ALTER DATABASE CLEAR UNARCHIVED LOGFILE UNRECOVERABLE DATAFILE
Parameters	none
Commands	ALTER DATABASE CREATE DATAFILE RECOVER DATABASE RECOVER TABLESPACE RECOVER DATAFILE

6

Incomplete Oracle Recovery with Archiving

Objectives

Objectives

After completing this lesson, you should be able to do the following:

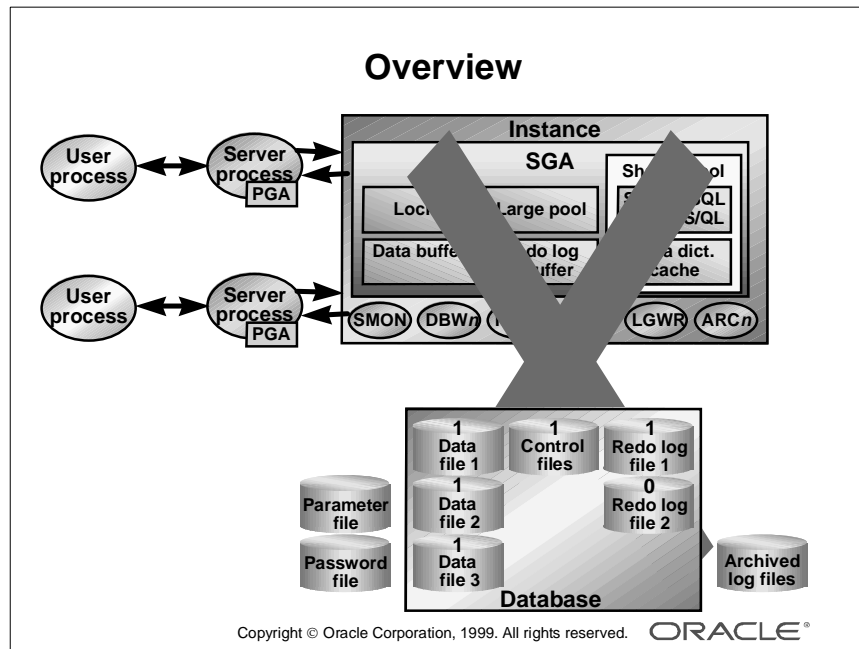
- **Determine when to use an incomplete recovery to recover the system**
- **Perform an incomplete database recovery**
- **Recover after losing current and inactive nonarchived redo log files**

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Objectives

This lesson discusses how to perform incomplete recovery after losing database files while the database is being archived.

Overview



Overview

This lesson only discusses recovery situations for databases in ARCHIVELOG mode that need to be recovered before the time of the failure.

Incomplete Recovery

Reconstructs the database to a prior point-in-time (before the time of the failure).

Note: This situation results in the loss of data from transactions committed after the time of recovery. This data will need to be reentered manually. Only perform this recovery when absolutely necessary. Incomplete recovery can be a difficult and time-consuming operation.

Performing Incomplete Recovery

- Requires a valid offline or online backup of all the database files.
- Needs all archived logs from the backup until the specified time of recovery.
- Used when a severe failure occurs, such as:
 - A failed complete recovery operation
 - Important tables in the database are accidentally dropped

Reasons for Incomplete Recovery

Reasons for Incomplete Recovery

- **User error**
 - An important table was dropped.
 - Bad data was committed in a table.
- **Complete recovery fails because an archived log is lost.**
- **Loss of all control files**
- **Loss of all unarchived redo log files and a data file.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Common Causes

- **User error:** For example, a user drops the wrong table, commits data updated with an incorrect WHERE clause, and so forth.
- **Missing archive:** For example, a complete recovery operation fails because of a bad or missing archived log. Recovery can only be completed to a time in the past, prior to applying the archived log.
- **Loss of control files:** For example, you did not mirror your control file, you do not know the structure of your database, but you have a backup of an old binary copy.
- **Loss of redo logs:** For example, redo logs were not mirrored and you lost a redo log before it was archived, along with a data file. Recovery cannot continue past the lost redo log.
- **Distributed databases:** Incomplete recovery at one location requires incomplete recovery of all other databases on the distributed network.

Incomplete Recovery

Types of Incomplete Recovery

- **Time-based recovery**
- **Cancel-based recovery**
- **Recovery using a backup control file**
- **Change-based recovery**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Types of Incomplete Recovery

These types of Incomplete Recovery use the RECOVER DATABASE command.

Time-Based Recovery This method of recovery is terminated after the database has committed all changes up to the specified point-in-time. Use this approach when:

- Unwanted changes to data were made or important tables dropped, and the approximate time of the error is known. Recovery time and data loss will be minimized if you are notified immediately. Well tested programs, security, and procedures should prevent the need for this type of recovery.

Note: Oracle8i supplies a new utility called LogMiner that performs granular logical recovery by undoing specific changes (DML operations) made by one or more transactions. This may minimize the need for performing point-in-time recovery to recover from a logical application error and provides the exact time of the error occurrence.

- The approximate time a nonmirrored online redo log becomes corrupt. Mirroring of logs should prevent the need for this type of recovery.

Note: In this situation, LogMiner may provide the exact time of the corruption.

Types of Incomplete Recovery (continued)

Cancel-Based Recovery This method of recovery is terminated by entering CANCEL at the recovery prompt (instead of a log file name). Use this approach when:

- A current redo log file or group is damaged and is not available for recovery. Mirroring should usually prevent the need for this type of recovery.
- Archived log needed for recovery is lost. Frequent backups and multiple archive destinations should prevent the need for this type of recovery.

Recovery Using a Backup Control File This method of recovery is terminated when the specified method of recovery (cancel, time, or change) has completed or control files are recovered. You must specify in the RECOVER DATABASE command that an old copy of the control file will be used for recovery. Use this approach when:

- All control files are lost, the control file cannot be recreated, and a binary backup of the control file exists. Mirroring the control file (onto different disks) and keeping a current text version of the CREATE CONTROLFILE statement will reduce the chances of using this method.
- Restoring a database, with a different structure to the current database, to a prior point-in-time.

Change-Based Recovery This method of recovery is terminated after the database has committed all changes up to the specified system change number (SCN). Use this approach when recovering databases in a distributed environment. This method is not described in more detail in this course. Attend the *Oracle8i: Distributed Database* class for more information.

RECOVER Command

RECOVER Command Syntax

Recover a database until cancel:

```
SQL> recover database until cancel;
```

Recover a database until time:

```
SQL> recover database
2 until time '1999-03-04:14:22:03';
```

Recover using backup control file:

```
SQL> recover database
2 until time '1999-03-04:14:22:03';
3 using backup controlfile;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Recover Command

The following command is used to perform incomplete recovery:

```
recover [automatic] database <Option>
```

where: **automatic** Automatically applies archived and redo log files.

Option until time 'YYYY-MM-DD:HH:MI:SS';
 until cancel;
 until scn <integer>;
 using backup control file;

Note

- The ALTER DATABASE RECOVER syntax may be used instead.
- To apply redo log files automatically during recovery, you can use the SQL*Plus command SET AUTORECOVERY ON, enter `auto` at the recovery prompt, or use the SQL command RECOVER AUTOMATIC.

Recovery Steps

Recovery Steps

1. Shut down and back up the database.
2. Restore all data files.
3. Do not restore the control file, redo logs, password files, or parameter files.
4. Mount the database and recover the data files before the time of failure.
5. Open the database with RESETLOGS.
6. Perform a closed database backup.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Incomplete Recovery

When a failure occurs requiring incomplete recovery (only with a database in ARCHIVELOG mode), you must have the following in order to recover:

- A valid offline or online backup containing all data files.
- All archived logs, from the restored backup to before time of the failure.

How to Recover

If you meet the above requirements for incomplete recovery, then perform these steps to recover:

- 1 Perform a full closed backup of the existing database.
- 2 Make sure that the database is shut down first, since all data files will be restored from a previous backup (including system data files).
- 3 Restore all data files to take your database back in time.
- 4 Place the database in mount mode and recover the database.
- 5 Open the database by using the RESETLOGS option and make sure the database problem has been fixed (if not, follow the recover steps again).
- 6 Perform a whole closed backup of the database.

Incomplete Recovery Guidelines

Recovery Guidelines

- **Follow all steps: Most errors occur during this type of recovery.**
- **Whole-database backups before and after recovery assist future recovery.**
- **Always verify that the recovery is successful.**
- **Back up the control file regularly.**
- **Back up and remove archived logs.**
- **Databases are brought forward in time, not back in time.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Incomplete Recovery Guidelines

- It is important to follow all recovery steps, since most problems with recovery are caused by a DBA error during incomplete recovery.
 - Perform a whole closed database backup (including control files and redo logs) before starting incomplete recovery. This assists in two ways:
 - Recover from error. If your recovery fails (for example, you recover past the desired point of recovery), redo logs and control files cannot be used for the next recovery, unless there is a backup of these files.
 - Save time if recovery fails. In this situation, you can restore the data files from the new backup, rather than from a previous backup which needs archives applied.
- Note:** If a whole backup is not performed, at least archive the current redo log (ALTER SYSTEM ARCHIVE LOG CURRENT) and backup the control file (ALTER DATABASE BACKUP CONTROLFILE TO <LOCATION>).
- Perform a whole closed backup after successful recovery. This is recommended if a recovery is required before completion of the next scheduled backup.

Incomplete Recovery Guidelines (continued)

- Always verify that the failure has been fixed before enabling users back on to the system, in case the recovery fails and needs to be performed again.
 - The failure can happen if a user notifies you that the table was dropped at 11:45 a.m.
 - You recover to 11:44 a.m. (before the table was dropped), only to find after recovery that the table still is not there.
 - You find that the user's watch is five minutes fast. You should have restored to 11:39 a.m.

Note: Use the LogMiner utility to get the exact time of the error occurrence (Refer to the lesson "Oracle Utilities for Troubleshooting").

- Make a control file backup whenever the database structure changes, since a backup control file must be used if the current database structure is different from the structure at the time you want to recover to.
- After an incomplete recovery, the database must be opened with the RESETLOGS option to synchronize all database files. During this command, if there are any missing redo log files, they are automatically recreated.
- Back up (and later remove) archived logs from the system to prevent mixing archives from different database incarnations.
 - For example, a database at log seq 144 has archived logs from `arch_120.rdo` to `arch_143.rdo`.
 - After performing incomplete recovery, a new database incarnation is created, setting the database log seq to 0.
 - Archived logs `arch_120.rdo` to `arch_143.rdo` are now part of the old database incarnation.
 - After 120 log switches, the archived log `arch_120.rdo` will be overwritten, and is backed up with all other archives (including the old archived logs `arch_121.rdo` to `arch_143.rdo`).
 - At a later stage, if recovery requires `arch_124.rdo`, you need to make sure that the archived log restored from the backup is for the correct database incarnation, otherwise an error will result.
- Transaction activity can only be rolled forward to the desired time, not back to the desired time. This is the reason why all data files must be restored for the database to be taken back in time. Failure to restore all data files will prevent the (unsynchronized) database from opening.

Note: If the LogMiner utility provides enough information for performing granular logical recovery by undoing specific changes made by one or more transactions, then this may sometimes prevent the need for performing point-in-time recovery to recover from a logical application error. (Refer to the lesson "Oracle Utilities for Troubleshooting").

- The RECOVER DATABASE command is usually easier to use than the ALTER DATABASE.

The Alert Log

Checking the Alert Log

- Check the alert log before and after recovery.
- Save and clear the log frequently.
- Information is recorded during recovery.
- Information is stored in **BACKGROUND_DUMP_DEST**.
- The filename is **alert_<SID>.log**.
- The alert log is useful for finding recovery errors, hints, and SCNs.

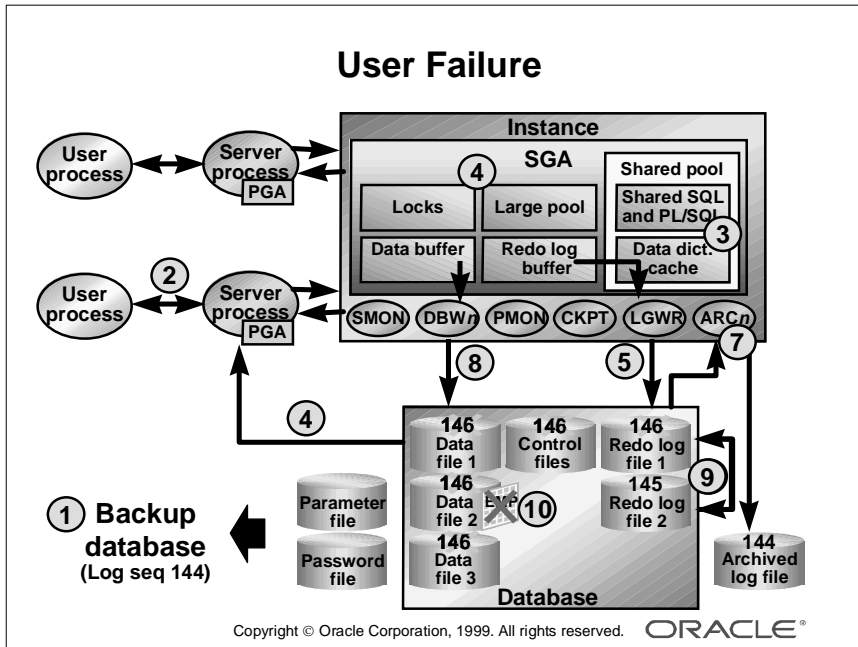
Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

The Alert Log

During recovery, progress information is stored in the alert log. This file should always be checked before and after recovery. For example:

```
$ vi /disk1/BDUMP/alert_DB00.log
...
Media Recovery Log
ORA-279 ... RECOVER    database until time '1999...
Wed Mar 09 11:55:13 1999
RECOVER DATABASE CONTINUE DEFAULT
Media Recovery Log /disk1/archive/arch_34.rdo
Incomplete recovery done UNTIL CHANGE 309121
Media Recovery Complete
Completed: RECOVER DATABASE CONTINUE DEFAULT
Wed Mar 09 11:55:13 1999
alter database open resetlogs
...
```

Time-Based Recovery



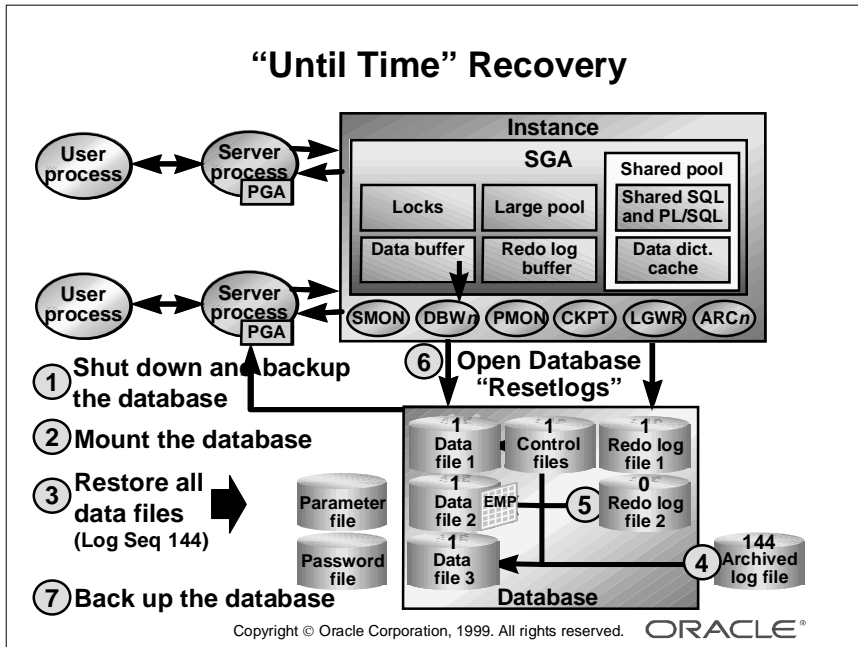
Time-Based Recovery

Scenario

- **The current time is 12:00 p.m. on 9-Mar-99.**
- **Your training DBA just told you he dropped the employee (EMP) table.**
- **The table was dropped around 11:45 a.m.**
- **Database activity is minimal because most staff are currently in a meeting.**
- **The table must be recovered.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE[®]

Incomplete Recovery Using Until Time



Incomplete Recovery Using Until Time

Shut down the database and begin recovery. Since the approximate time of the failure is known and the database structure has not changed since 11:44 a.m., we can use the “until time” method:

Step	Explanation
1	If the database is opened, shut it down by using either the NORMAL or IMMEDIATE or TRANSACTIONAL options.
2	Mount the database.
3	Restore all data files from backup (the most recent if possible): <pre>UNIX> cp /disk1/backup/*.dbf /disk1/data/</pre> <pre>UNIX> cp /disk2/backup/*.dbf /disk2/data/</pre> <pre>UNIX> ...</pre> <pre>NT> copy c:\backup*.dbf c:\data\</pre> <pre>NT> copy d:\backup*.dbf d:\data\</pre> <pre>NT> ...</pre>
4	You may need to restore archived logs. If there is enough space available, restore to the LOG_ARCHIVE_DEST location or use the ARCHIVE SYSTEM ARCHIVE LOG START TO <LOCATION> or SET LOGSOURCE <LOCATION> to change the location.

Step	Explanation				
5	<p>Recover the database:</p> <pre>SQL> recover database until time '1999-03-09:11:44:00';</pre> <p>ORA-00279: change 148448 ...02/29/98 17:04:20 needed for thread</p> <p>ORA-00289: suggestion : /disk1/archive/arch_6.rdo</p> <p>ORA-00280: change 148448 for thread 1 is in sequence #6</p> <p>Log applied.</p> <p>...</p> <p>Media recovery complete.</p>				
6	<p>To synchronize data files with control files and redo logs, open database by using RESETLOGS option:</p> <pre>SQL> alter database open resetlogs;</pre> <pre>SQL> archive log list;</pre> <p>...</p> <p>Oldest online log sequence 0</p> <p>Next log sequence to archive 1</p> <p>Current log sequence 1</p>				
7	<p>Before performing the whole closed database backup, query the EMP table to make sure it exists. If you get the following error:</p> <p>ORA-00942: table or view does not exist</p> <p>then locate the schema for the table by using:</p> <pre>SQL> select table_name, owner 2 from dba_tables where table_name = 'EMP';</pre> <table> <thead> <tr> <th>TABLE_NAME</th><th>OWNER</th></tr> </thead> <tbody> <tr> <td>EMP</td><td>PETER</td></tr> </tbody> </table> <p>1 row selected.</p> <p>Note: If no rows are selected, then the table has not been restored correctly. You will need to recover to an earlier point-in-time, otherwise perform the backup.</p> <p>Note: LogMiner can help you to find the right time of the DROP statement.</p>	TABLE_NAME	OWNER	EMP	PETER
TABLE_NAME	OWNER				
EMP	PETER				
8	<p>When recovery is successful and the backup has completed, notify users that the database is available for use, and any data entered after the recovery time (11:44 a.m.) will need to be reentered.</p>				

Incomplete Recovery Using Until Cancel

Cancel-Based Recovery

Scenario

- The current time is 12:00 p.m. on 9-Mar-99.
- Your training DBA dropped the EMP table while trying to fix bad blocks.
- Log files exist on the disk containing the employee data file.
- The table was dropped around 11:45 a.m.
- Staff are currently in a meeting.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Incomplete Recovery Using Until Cancel

You are concerned about block corruption in the EMP table resulting from disk error. Since redo logs are contained on the same disk, you decide to check the status of redo logs and archived logs:

```
SQL> select * from v$logfile;
```

GROUP#	STATUS	MEMBER
--------	--------	--------

2		/disk1/data/log2a.rdo
1		/disk1/data/log1a.rdo

```
SQL> select * from v$log;
```

G#	...SEQ#	BYTES	MEMBERS	ARC	STATUS	... FIRST_TIME
1	... 49	153600	1	NO	CURRENT	... 99-03-09:11:55
2	... 48	153600	1	NO	INACTIVE	... 99-03-09:11:34

Cancel-Based Recovery

Findings

- Redo logs are not multiplexed.
- One of the online redo logs is missing.
- The missing redo log is not archived.
- The redo log contained information from 11:34 a.m.
- Twenty-six minutes of data will be lost.
- Users can recover their data.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Incomplete Recovery Using Until Cancel (continued)

After searching through the `/disk1/data` directory, you notice that redo log `log2a.rdo` cannot be located and has not been archived. You therefore cannot recover past this point. Querying `V$LOG_HISTORY` confirms the absence of archived log seq 48 (`log2a.rdo`):

```
SQL> select * from v$log_history;
```

RECID	STAMP	...	FIRST_CHANGE	FIRST_TIME
-----	-----	...	-----	-----
1	318531466	...	88330	99-02-28:12:43
47	319512880	...	309067	99-03-09:11:26

Since this is an OLTP system, the output from `V$LOG` shows an extra 10 minutes work will be lost if the database is recovered before applying `log2a.rdo`. Users are not happy about losing work, but can recover that work. You therefore start restoring the database:

- 1 Shut down the database.
- 2 You already have a valid backup, so mount the instance.
- 3 Restore all data files from the most recent backup.

Incomplete Recovery Using Until Cancel (continued)

4 Recover the database until log seq 48:

```
SQL> recover database until cancel
ORA-00279: change 148448 ...03/02/99 12:45:20 needed for thread
ORA-00289: suggestion : /disk1/archive/arch_34.rdo
ORA-00280: change 148448 for thread 1 is in sequence #34
Log applied.
...
ORA-00279: change 309012...03/09/99 11:33:56 needed for thread 1
ORA-00289: suggestion : /disk1/archive/arch_48.rdo
ORA-00280: change 309012 for thread 1 is in sequence #48
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
cancel
Media recovery cancelled.
```

5 Open the database by using the RESETLOGS option.

6 Check that the EMP table exists by performing a simple select.

7 When recovery is successful, notify users that the database is available for use, and any data entered after the recovery time (11:34 a.m.) will need to be reentered.

Incomplete Recovery Using the Backup Control File

Backup Control File Recovery

Scenario

- The current time is 12:00 p.m. on 9-Mar-99.
- Your training DBA dropped the entire employee tablespace, not just the table.
- The error occurred around 11:45 a.m.
- Many employee records were updated this morning, but not since 11 a.m.
- Backups are taken every night.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Incomplete Recovery Using the Backup Control File

- 1 Your training DBA removed the EMP table with this command:

```
SQL> drop tablespace emp_ts including contents;
```

- 2 You immediately ask users to log out and keep records of the data entered over the past 15 minutes. While you wait for all users to log out and remaining sessions to be killed, you place the database in restricted mode to prevent further access:

```
SQL> alter system enable restricted session;
```

- 3 During your investigation, you locate the binary control file from the backup last night. Since the current control file will be replaced, you carefully gather database structure information in case it is required:

```
SQL> select * from v$log;
```

GROUP#...	SEQ#	BYTES ...	ARC	STATUS	... FIRST_TIME
1 ...	61	153600...	NO	CURRENT	... 99-03-09:11:55
2 ...	60	153600...	NO	INACTIVE	... 99-03-09:11:34

```
SQL> select tablespace_name, file_name from dba_data_files
```

```
2 where tablespace_name = 'EMP_TS';
```

TABLESPACE_NAME	FILE_NAME
EMP_TS	/disk1/data/emp_01.dbf

Backup Control File Recovery

Findings

- The backup last night contains data files and control files required for recovery.
- Employee tablespace has one data file.
- The current log sequence number is 61.
- You confirm that the tablespace was dropped at 11:44:54 a.m. on 9-MAR-99.
- Data file number 4 is offline.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Incomplete Recovery Using the Backup Control File (continued)

- 4 You confirm the time of error by checking the alert log:

```
UNIX> vi /disk1/BDUMP/alert*.log
or NT> notepad c:\BDUMP\alert_DB00.log
...
Wed Mar09 11:44:54 1999
drop tablespace emp_ts including contents
...
```

- 5 Shut down the database, back up control files, then restore all data files and control files for the database at a time when the tablespace existed. After attempting to open the database, the following error informs you that the redo logs and control files are not synchronized:

```
ORA-00314: log 1 of thread 1, expected sequence# doesn't match
ORA-00312: online log 1 thread 1: '/disk1/data/log1a.rdo'
```

Incomplete Recovery Using the Backup Control File (continued)

- 6** Verify whether any offline data files exist and place them online, since any offline files may be unrecoverable after recovery:

```
SQL> select * from v$recover_file;
FILE#      ONLINE      ERROR      CHANGE#    TIME
-----      -
         4      OFFLINE                288772    02-MAR-99
SQL> alter database datafile 4 online;
```

- 7** Perform the recovery:

```
SQL> recover database until time '1999-03-09:11:44:00' \
      2 using backup controlfile;
...
Media recovery complete.
```

Note: If the following error appears instead of Media recovery complete, it indicates that either data files need to be restored from an earlier backup, or more recovery is required (not possible here).

```
ORA-01152: file 7 was not restored from a sufficiently old backup
ORA-01110: datafile 7: '/disk1/data/new01.dbf'
```

- 8** To synchronize data files with the control files and redo logs, open the database with the RESETLOGS option.
- 9** Verify that the EMP table exists by performing a simple select.
- 10** When recovery is successful, notify users that the database is available for use, and any data entered after 11:44 a.m. will need to be reentered.

Loss of Current Online Redo Logs

Loss of Current Redo Logs

If database is open:

- Determine the current log group.
- Use the **CLEAR LOGFILE** command.
- Database becomes operational.
- Perform a whole-database backup.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Loss of Current Redo Logs

There are two possible scenarios when the current online redo log is lost:

- Database is opened, but in a hung state.
- Database is closed due to media failure or a terminated background process.

Database Opened, But in a Hung State

If the database is opened, the file might be corrupt or accidentally deleted. Use the steps below to rectify the hung database problem:

1 Determine which group is current:

```
SQL> select * from v$log;
```

GROUP#...	SEQ#	BYTES ...	ARC	STATUS	... FIRST_TIME
-----...	----	-----...	---	-----	... -----
1 ...	61	153600...	NO	CURRENT	... 99-03-09:11:55
2 ...	60	153600...	NO	INACTIVE	... 99-03-09:11:34

Database Opened, But in a Hung State (continued)

- 2** Clear the current log file using the following command:

```
SQL> alter database clear unarchived logfile group 1;
```

- 3** The database should now be operational, since the logfile will be over-written in the case of corruption, or recreated in the case of file loss.
- 4** If the database crashes due to media failure before the next backup, then incomplete recovery will be required since recovery information has just been cleared. You should therefore immediately perform a closed whole backup.

Loss of Current Redo Logs

If database is closed:

- **Attempt to open the database.**
- **Find the current log sequence number.**
- **Recover the database until cancel.**
- **Drop and re-create log files if necessary.**
- **Open the database using RESETLOGS.**
- **Perform a whole-database backup.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Database Closed Because of Failure

If the database is closed, media failure may have occurred or a background process may have terminated. Use the steps below to rectify this situation:

- 1** Attempting to open the database will immediately notify you of the current redo log group through the following message:

```
Database mounted.
```

```
ORA-00313: open failed for members of log group 2 of thread 1
```

```
ORA-00312: online log 2 thread 1: '/disk1/archive/log2a.rdo'
```

```
ORA-27037: unable to obtain file status
```

```
SVR4 Error: 2: No such file or directory
```

```
Additional information: 3
```

Because log group 2 is the current log group, it will not have been archived. Using the CLEAR LOGFILE command is no use because:

```
SQL> alter database clear unarchived logfile group 2;
```

```
ORA-01624: log 2 needed for crash recovery of thread 1
```

```
ORA-00312: online log 2 thread 1: 'disk1/archive/log2a.rdo'
```

Database Closed Because of Failure (continued)

- 2** Incomplete recovery is therefore required. First, you must note the current log sequence number:

```
SQL> select * from v$log;
```

GROUP#...	SEQ#	BYTES ...	ARC	STATUS	... FIRST_TIME
1 ...	61	153600...	NO	CURRENT	... 99-03-09:11:55
2 ...	60	153600...	NO	INACTIVE	... 99-03-09:11:34

- 3** Restore all data files from a previous backup and use recover until cancel to stop before redo log 61 is applied:

```
SQL> recover database until cancel;
```

```
ORA-00279: change 309043...03/09/99 14:50:14 needed for thread 1
```

```
ORA-00289: suggestion : /disk1/archive/arch_46.rdo
```

```
ORA-00280: change 309043 for thread 1 is in sequence #46
```

```
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

```
...
```

```
ORA-00279: change 309141...03/09/99 19:50:14 needed for thread 1
```

```
ORA-00289: suggestion : /disk1/archive/arch_61.rdo
```

```
ORA-00280: change 309043 for thread 1 is in sequence #61
```

```
ORA-00278: log file ... no longer needed for this recovery
```

```
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

```
cancel
```

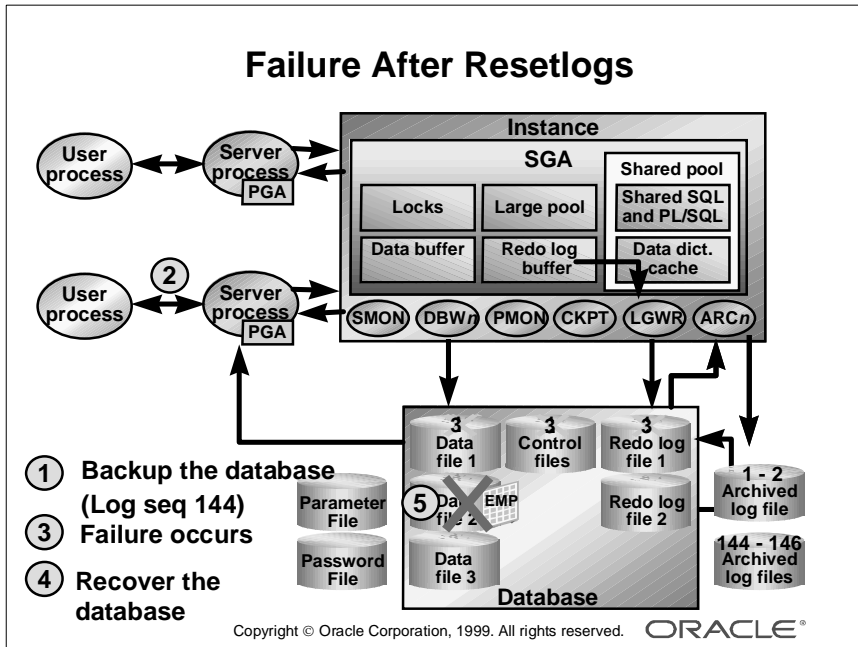
```
Media recovery complete.
```

- 4** Open the database using the RESETLOGS option.
- 5** The database should now be operational, since any missing logfiles will be recreated.

Note: If logfiles need to be recreated on another disk due to media failure, use the ALTER DATABASE DROP LOG GROUP and ALTER DATABASE ADD LOG GROUP commands to create the log files manually.

- 6** Since you just performed incomplete recovery, the database should now be backed up.

Recovery Through Resetlogs



Recovery Through Resetlogs

Use this process only when:

- The database version is 7.3.3 and later
- There is no backup after RESETLOGS
- There is a backup prior to RESETLOGS
- Control files exist before and after RESETLOGS
- Archived and redo logs are available.
- The alert log contains RESETLOGS information.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

When to Use This Process

By making backups after every incomplete recovery, this procedure should never have to be performed. However, large databases with 24-hour, 7-days a week availability require reduced recovery time. Therefore, opened database backups after recovery are enforced, and if media failure occurs before the backup is completed, this recovery might be required.

Prerequisites

In order to use this method, you must meet the following criteria:

- The database currently must be an Oracle RDBMS version 7.3.3 (or higher) as well as before the RESETLOGS.
- There is no whole closed or opened database backup after RESETLOGS.
- There is a whole closed or opened database backup prior to RESETLOGS.
- Control files before and after RESETLOGS exist.
- All archived logs and online redo logs are present for recovery.
- The alert log contains information from the previous RESETLOGS.

Procedure

Follow *every* step in this procedure. For help, contact Oracle Worldwide Support.

- 1 Copy the current database control files to another location.
- 2 Restore the data files and control files from a backup before the last RESETLOGS operation.
- 3 Mount the database.
- 4 From the alert log, locate the change number calculated at the completion of the previous RESETLOGS operation.

...

```
Incomplete recovery done UNTIL CHANGE 309121
```

```
Media Recovery Complete
```

...

- 5 Recover the database until the change number located above:

```
SQL> recover database until change 309121 \
```

```
2 using backup controlfile;
```

```
ORA-00279: change 309043...03/09/99 14:50:14 needed for thread 1
```

```
ORA-00289: suggestion : /home/disk1/user4/ARCHIVE/arch_46.rdo
```

```
ORA-00280: change 309043 for thread 1 is in sequence #46
```

```
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

```
auto
```

...

```
Media Recovery Complete.
```

- 6 Shut down the database using the “normal” option.
- 7 Restore the current control files from the location specified in step 1.
- 8 Mount the database.
- 9 Recover the database using the appropriate option in RECOVER DATABASE command.
- 10 Open the database.
- 11 Validate the data you were trying to recover.
- 12 View V\$LOG to make sure that recovery went to the correct log sequence number.
- 13 Back up the database to prevent recovery through reset logs.

Summary

Summary

In this lesson, you should have learned how to:

- Use time-based recovery when the time of error is known
- Use cancel-based recovery for lost archived logs or current online redo logs
- Back up the control file when current and backup database structures differ
- Use the RESETLOGS option when opening the database
- Take backups before and after recovery

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Quick Reference

Context	Reference
Parameters	BACKGROUND_DUMP_DEST
Dynamic performance views	V\$LOG V\$LOGFILE V\$LOG_HISTORY V\$RECOVER_FILE
Data dictionary views	DBA_DATA_FILES
Commands	ALTER DATABASE OPEN RESETLOGS ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP n ALTER DATABASE DATAFILE [OFFLINE ONLINE] ALTER SYSTEM ENABLE RESTRICTED SESSION ALTER SYSTEM ARCHIVE LOG START TO <location> ALTER SYSTEM ARCHIVE LOG CURRENT RECOVER DATABASE UNTIL [CANCEL TIME CHANGE] RECOVER DATABASE UNTIL TIME [USING BACKUP CONTROLFILE]

Oracle Export and Import Utilities

Objectives

Objectives

After completing this lesson, you should be able to do the following:

- Use the Export utility to create:
 - A complete logical backup of a database object
 - An incremental backup of a database object
- Invoke the direct-path method export
- Use the Import utility to recover a database object
- Perform a tablespace point-in-time recovery (TSPITR) with or without the transportable tablespace feature

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Export and Import Utility Overview

Oracle Export and Import Utility Overview

These utilities enable you to do the following:

- Archive historical data
- Save table definitions (with or without data) to protect from user error failure
- Move data between machines and databases or versions of the Oracle server
- Transport tablespaces between databases
- TSPITR

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Export and Import Utility Overview

- The Export utility provides a logical backup of
 - A database object
 - Schema's objects
 - A tablespace
 - An entire database
- The Import utility is used to read a valid Export file for moving data into a database. Redo log history can not be applied to objects that are imported from an export file, therefore data loss may occur but can be minimized.

The DBA can use the Export and Import utilities to compliment normal operating system backups by using them to:

- Create a historical archive of a database object or entire database. For example, when a schema is modified to support changing business requirements.
- Save table definitions in a binary file. This may be useful for creating a to maintain a baseline of a given schema structure.
- Move data from one Oracle version to another, such as upgrading from Oracle7 to Oracle8.

Export and Import Utility Overview (continued)

- Protect against:
 - User errors where a user may accidentally drop or truncate a table
 - A table that has become logically corrupted
 - An incorrect batch job or other DML statement that has affected only a subset of the database
- Recover:
 - A logical database to a point different from the rest of the physical database when multiple logical databases exist in separate tablespaces of one physical database
 - A tablespace in a Very Large Data Base (VLDB) when tablespace point-in-time recovery (TSPITR) is more efficient than restoring the whole database from a backup and rolling it forward

Note: This lesson addresses the Export and Import utilities and discusses how they affect backup and recovery operations. For a detailed description of these utilities, refer to the *Oracle8i Server Utilities* manual.

Methods to Run the Export Utility

Methods to Run the Export Utility

- An interactive dialog
- The export page of the Data Manager within Enterprise Manager
- The command line interface, by specifying parameters

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Export Methods

- Interactive dialog: By specifying the EXP command at the operating system and no parameters, the Export utility will prompt you for inputs, supplying default values.
- The export page of Data Manager within Oracle Enterprise Manager
- If command line mode is chosen, the selected options must be explicitly specified on the command line. Any missing options will default to Export utility default values.

Note: Many options are only available by using the command line interface. However, you can use a parameter file with command line.

Export Modes

Export Modes			
Table Mode	User Mode	Tablespace Mode	Full Database Mode
Table definitions	Tables definitions	Table definitions	Tables definitions
Table data (all or selected rows)	Tables data		Tables data
Owner's table grants	Owner's grants	Grants	Grants
Owner's table indexes	Owner's indexes	Indexes	Indexes
Table constraints	Tables constraints	Table constraints	Tables constraints
		Triggers	

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Table Mode

Table mode exports specified tables in the user's schema, rather than all tables. A privileged user can export specified tables owned by other users.

User Mode

User mode exports all objects for a user's schema. Privileged users can export all objects in the schemas of a specified set of users. This mode can be used to compliment a full database export.

Tablespace Mode

You can use transportable tablespaces to move a subset of an Oracle database and plug it into another Oracle database, essentially moving tablespaces between the databases. Moving data by way of transportable tablespaces can be much faster than performing either an import/export of the same data, because transporting a tablespace only requires the copying of data files and integrating the tablespace structural information. You can also use transportable tablespaces to move index data, thereby avoiding the index rebuilds you would have to perform when importing table data.

Full Database Mode

Full database mode exports all database objects, except those in the SYS schema. Only privileged users can export in this mode.

Command Line Export

Command Line Export

Syntax

```
exp keyword = (value, value2, ... ,valuen)
```

Example

```
exp scott/tiger TABLES=(emp,dept) rows=y
file=expincr1.dmp
```

```
exp system/manager OWNER=SCOTT direct=y
```

```
exp system/manager TRANSPORT_TABLESPACE=y
TABLESPACES=(ts_emp) log=ts_emp.log
```

```
exp system/manager FULL=y inctype=cumulative
file=expcum1.dmp
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Command Line Export

You can copy database data to an operating system file by using the command line mode of the Export utility. This file is only readable by the Import utility.

Example

Create an export file named `expincr1.dmp` that includes the tables, `emp` and `dept`, for `scott`'s schema including rows:

```
$ exp scott/tiger tables=(emp,dept) rows=y file=expincr1.dmp
```

Create a fast export file named `expdat.dmp` that includes all objects for `scott`'s schema including rows:

```
$ exp system/manager owner=SCOTT DIRECT=Y
```

Create an export file named `expdat.dmp` that includes all definitions of objects belonging to the tablespace `ts_emp` and generate a log file named `ts_emp.log`:

```
$ exp system/manager TRANSPORT_TABLESPACE=y TABLESPACES=(ts_emp)
LOG=ts_emp.log
```

Create an export file named `expcum1.dmp` that includes all definitions and data modified in the database since the last cumulative or complete export.

```
$ exp system/manager FULL=y INCTYPE=cumulative FILE=expcum1.dmp
```

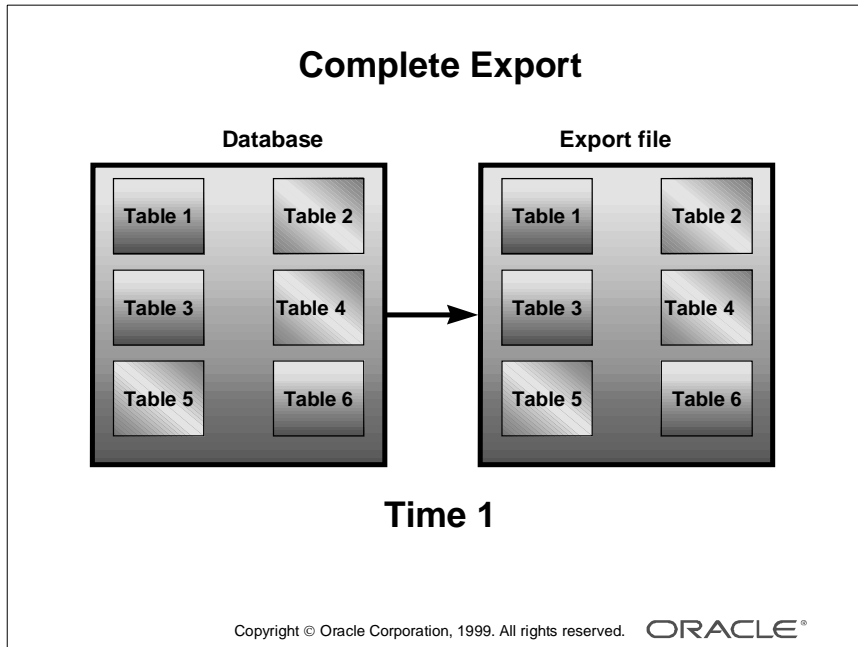
Note: Command line mode options are similar to interactive mode options.

Export Parameters

Parameter	Description
USERID	Username/password of schema objects to export
FILE	Name of output file
ROWS	Include table rows in export file: (Y)es/(N)o
FULL	Export entire database: (Y)es/(N)o
OWNER	Users to export: Username
TABLES	Tables to export: List of tables
INDEXES	Indexes to export: (Y)es/(N)o
DIRECT	Specify direct mode export: (Y)es/(N)o
INCTYPE	Type of export level
PARFILE	Name of file in which parameters are specified
HELP	Display export parameters in interactive mode (Y)
LOG	Name of file for informational and error messages
CONSISTENT	Read-consistent view of the database when data is updated during an export: (Y)es/(N)o
BUFFER	Size of the data buffer in bytes: (Integer)
TRANSPORT_TABLESPACE	Enables the export of transportable tablespace metadata (release 8.1 only)
TABLESPACES	Tablespaces to be transported (release 8.1 only)
POINT_IN_TIME_RECOVER	Indicates whether or not the Export utility exports one or more tablespaces in an Oracle database (release 8.0 only)
RECOVERY_TABLESPACES	Specifies the tablespaces that will be recovered by using point-in-time recovery (release 8.0 only) Refer to the Oracle Server Readme, release 8.0.4
COMPRESS	Specified to include all data in one extent: (Y)es/(N)o

Note: The parameters listed above are not a complete list of all Export utility parameters but those that a DBA may often use for restore purposes.

Complete Export



Complete Export

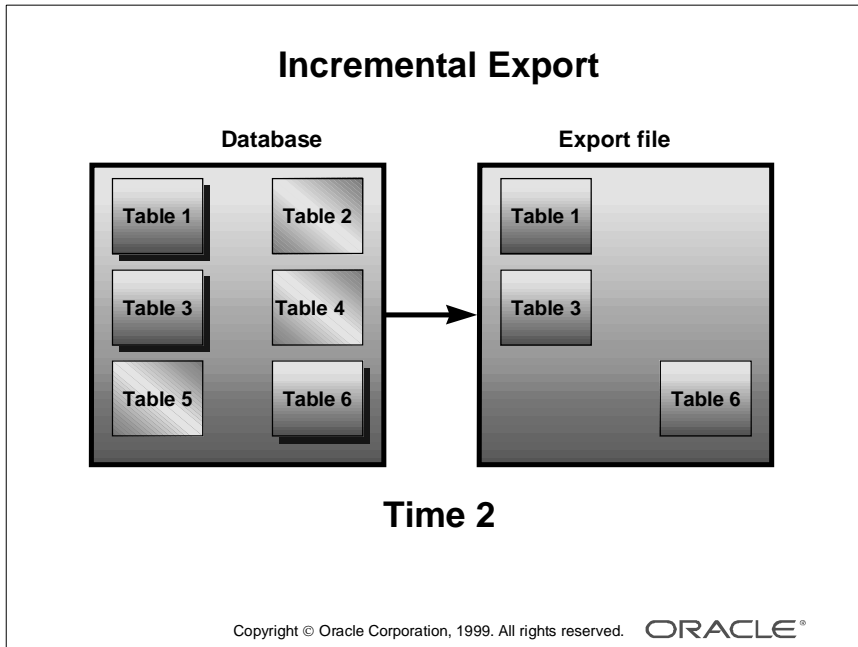
If you use cumulative and incremental exports, you should periodically perform a complete export to create a base backup. Following the complete Export, perform frequent incremental Exports and occasional cumulative Exports. After a given period of time, you should begin the cycle again with another complete Export.

As shown in the example above, a complete database export backs up all table and data definitions.

Restrictions

You can only perform complete, incremental or cumulative Exports only in full database mode (FULL=Y).

Incremental Export



Incremental Export

An incremental export includes objects that have changed since the last export of any kind. An incremental Export exports the table definition and all its data, *not just the changed rows*. Typically, you perform incremental Exports more often than cumulative or complete Exports.

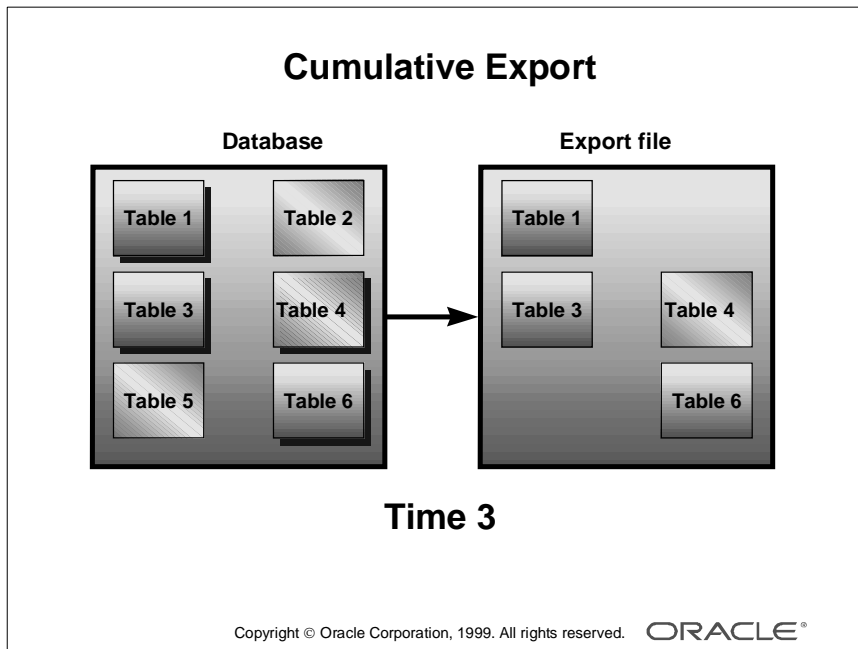
Any modification on (UPDATE, INSERT, or DELETE) on a table automatically qualifies that table for incremental Export.

Example

In the example, above, Tables 1, 3, and 6 incurred changes since the last Export at Time 1, which was a complete export.

Note: An incremental export may not be a very good strategy for applications that often access a few large tables. It is applicable for departmentalized applications where changes are more or less scattered across small tables.

Cumulative Export



Cumulative Export

A cumulative Export backs up tables that have changed since the last cumulative or complete Export.

Example

In the example above, Tables 1, 2, and 6 changed since Time 1 and Table 4 changed since Time 2. The cumulative Export therefore backs up all tables that changed since the last complete Export.

Incremental and Cumulative Export Benefits

Incremental and Cumulative Export Benefits

- Restore tables accidentally dropped by users
- Smaller export files
- Less time to export

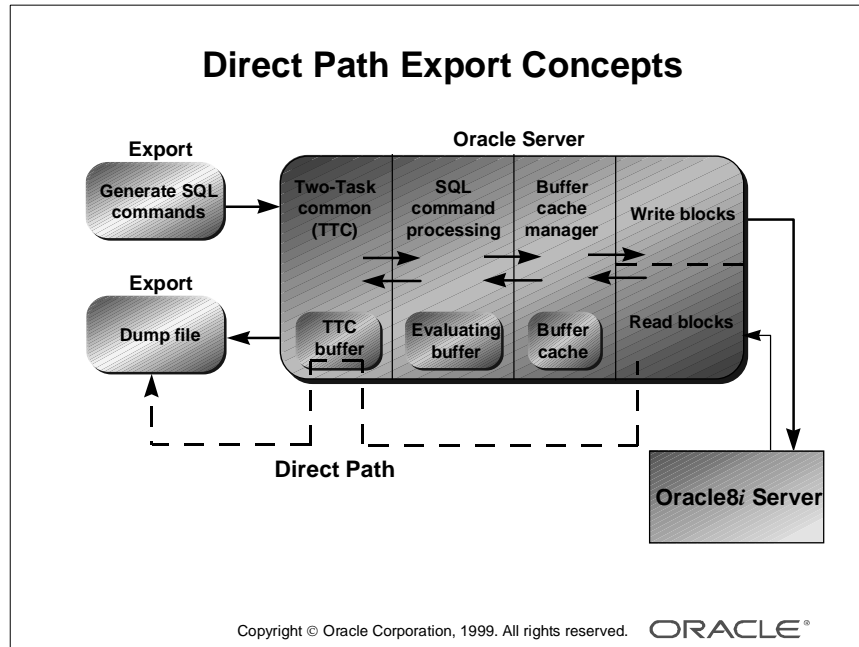
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Incremental and Cumulative Export Benefits

Incremental and cumulative Exports help solve the problems faced by DBAs who work in environments where many users create their own tables. The benefits of these type of Exports include:

- DBA restoration of tables accidentally dropped by users
- Smaller export files because less data is being exported
- Less time to export because only those objects that have changed since the last incremental or cumulative export are backed up

Direct Path Export Concepts



Direct Path Export Concepts

By using the Direct-Path feature, you can extract data much faster. When parameter `DIRECT=Y` is specified, Export utility reads directly from the data layer instead of going through the SQL-command processing layer.

Mechanics of Direct-Path Export

- Direct mode of export can be set by specifying the parameter `DIRECT=Y`.
- Direct-path export does not compete with other resources of the Instance.
- If in direct read mode, it reads database blocks into a private area used by the session.
- Rows are transferred directly into the Two-Task Common (TTC) buffer for transport.
- The data in the TTC buffer is in the format that the Export utility expects.

Specifying Direct Path Export

Specifying Direct Path Export

- As command line argument to the Export command:

```
exp userid=scott/tiger full=y direct=true
```
- As a keyword in a parameter file:

```
exp parfile=<Parameter file>
```

Parameter file
.....(Other Paramaters)
DIRECT = Y
.....(Other Paramaters)

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Specifying Direct Path Export

Before you can use direct-path Export, you must run the *catexp.sql* script.

Methods of by Using the DIRECT Parameter

Command-Line Option

You can invoke direct-path export by using the DIRECT command line parameter at the operating system prompt.

```
$ exp user=scott/tiger full=y direct=y
```

Parameter File

An example of a parameter file, *exp_par.txt*:

```
USERID=scott/tiger
TABLES=(emp,dept)
FILE=exp_one.dmp
DIRECT=Y
```

To execute the parameter from the operating system prompt:

```
$ exp parfile=exp_param.txt
```

Direct Path Export Features

Direct Path Export Features

- The type of export is indicated on the screen output, export dump file, and the log file.
- Data is already in the format that Export expects, avoiding unnecessary data conversion.
- Uses an optimized SQL SELECT statement.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Direct Path Export

The direct-path option of the Export utility introduces some features that differentiate it from the conventional-path export.

Direct-Path Features

- The type of export is indicated on the screen output, export dump file, and the log file specified with the LOG parameter.
- Data is already in the format that Export expects, thus avoiding unnecessary data conversion. The data is transferred to Export client, which then writes the data into the export file.
- The direct-path export uses an optimized SELECT * FROM *table* without any predicate.

Note: The format of the column data and specification in the export dump file differ from those of conventional-path export.

Direct Path Export Restrictions

Direct Path Export Restrictions

- The direct-path option cannot be invoked interactively.
- Client-side and server-side character sets must be the same.
- The BUFFER parameter has no affect.
- You cannot use the direct-path option to export rows containing LOG, BFILE, REF, or object types.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Direct-Path Restrictions

The direct-path option of the Export utility has some restrictions that differentiate it from the conventional-path export.

- The direct-path export feature cannot be invoked by using an interactive EXP session.
- When the direct-path option is used, the client-side character set must match the character set of the server side. Use the environment variable NLS_LANG to set the same character set as the server one.
- The BUFFER parameter of the EXPORT utility has no effect on direct-path Export; it is used only by the conventional-path option.
- You cannot direct-path Export rows that contain the datatypes of LOB, BFILE, REF, or object type columns, including VARRAY columns and nested tables. Only the data definition to create the table is exported, not the data.

Export Utility Compatibility Issues

Export Utility Compatibility Issues

- You can use the Export utility from any Oracle7 release to export from an Oracle8 server.
- The Oracle6 (or earlier) Export feature cannot be used against an Oracle8 database.
- Be aware of different versions.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Export Utility Compatibility

As a DBA, you need to be aware of the compatibility issues associated with the Export utility.

Issues

- You can use the Oracle7 Export utility against an Oracle8 database to create an Oracle7 export file.
- Oracle6 (or earlier) Export cannot be used against an Oracle8 database.
- When a lower version Export utility runs with a higher version of the Oracle server, categories of the database objects that did not exist in the lower version are excluded from the export.
- If a table was created by using the direct path option then the file is in a different format and can not be read by an Import utility prior to Oracle 7.3.
- Attempting to use a higher version of Export with Import of an earlier Oracle server often produces an error.
- You can use the transportable tablespaces only in an Oracle8i version database.

Using the Import Utility for Recovery

Uses of the Import Utility for Recovery

- **Create table definitions**
- **Extract data from a valid Export file**
- **Import from a complete, incremental, or cumulative export file**
- **Recover from user-error failures**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Import Utility

The Import utility can be used for recovery of data by using a valid Export utility file.

Uses of the Import Utility for Recovery

- Creating table definitions since the table definitions are stored in the export file. Choosing to import data without rows will create just the table definitions.
- Extracting data from a valid export file by using the Table, User, Tablespace, or Full Import modes.
- Importing data from a complete, incremental or cumulative export file.
- Recovering from user failure errors where a table is accidentally dropped or truncated by using one the previously mentioned methods.

Import Modes

Import Modes	
Mode	Description
Table	Import specified tables into a schema.
User	Import all objects that belong to a schema
Tablespace	Import all definitions of the objects contained in the tablespace
Full Database	Import all objects from the export file

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Table Mode

Table mode imports all specified tables in the user's schema, rather than all tables. A privileged user can import specified tables owned by other users.

User Mode

User mode imports all objects for a user's schema. Privileged users can import all objects in the schemas of a specified set of users.

Tablespace Mode

Tablespace mode allows a privileged user to move a set of tablespaces from one Oracle database to another.

Full Database Mode

Full database mode imports all database objects, except those in the SYS schema. Only privileged users can import in this mode.

Command Line Import

Command Line Import

Syntax

```
imp keyword = value or keyword = (value,  
value2, ... value n)
```

Example

```
imp scott/tiger TABLES=(emp,dept) rows=y  
file=expincr1.dmp
```

```
imp system/manager FROMUSER=scott  
file=expincr1.dmp
```

```
imp system/manager TRANSPORT_TABLESPACE=y  
TABLESPACES=ts_emp
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Examples

Import into scott's schema, the tables emp and dept, including rows, by using the export file named expincr1.dmp.

```
$ imp scott/tiger tables=(emp,dept) rows=y file=expincr1.dmp
```

Import all objects belonging to Scott schema, including rows, by using the export file named expincr1.dmp.

```
$ imp system/manager FROMUSER=scott file=expincr1.dmp
```

Import all definitions of objects belonging to the tablespace ts_emp, by using the export file named expdat.dmp.

```
$ imp system/manager TRANSPORT_TABLESPACE=y TABLESPACES=ts_emp
```

Note: Command line mode options are similar to interactive mode options.

Import Parameters

Parameter	Description
USERID	Username/password of schema objects to export
FILE	Name of the input file; must be a valid Export Utility file
ROWS	Include table rows in import file
IGNORE	Ignore create errors due to an object's existence
FULL	Import entire file
TABLES	Tables to import
INDEXES	Indexes to import
INCTYPE	Specifies the type of incremental import; options are SYSTEM and RESTORE
PARFILE	Parameter specification file
HELP	Display export parameters in interactive mode
LOG	File for informational and error messages
DESTROY	Specifies whether or not the existing data file making up the database should be reused
FROMUSER	A list of schemas containing objects to import
TOUSER	Specifies a list of usernames whose schemas will be imported
INDEXFILE	Specifies a file to receive index-creation commands
TRANSPORT_TABLESPACE	Instructs Import to import transportable tablespace metadata from an export file
TABLESPACES	List of tablespaces to be transported into the database
DATAFILES	List the datafiles to be transported into the database
TTS_OWNERS	List the users who own the data in the transportable tablespace set
POINT_IN_TIME_RECOVER	Indicates whether or not the Import utility recovers one or more tablespaces in an Oracle database to a prior point in time without affecting the rest of the database (release 8.0 only)

Note: The parameters listed above are not a complete list of all Import utility parameters but those that a DBA may often use for restore purposes.

Import Process Sequence

Import Process Sequence

1. **New tables and indexes are created.**
2. **Data is imported—indexes updated.**
3. **Triggers are imported, and integrity constraints are enabled on new tables.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Import Process Sequence

When importing a table, the export file is read and the table and data are created in the following order:

- 1 New tables are created. A key point is that these are technically NEW tables to the Oracle server. They have the same data and characteristics as the original tables, but in reality they are being newly created in the database. The reason that you cannot apply archive logs to roll forward an import is that there is no synchronizing information stored in either the logs or the export to specify when to start applying archived logs.
- 2 Index structures are built. The DBA can save some time during the import process by setting the parameter INDEXES=N, then building the indexes following the Import process. This will also limit the number of rollback segments required to support the import.
- 3 Data is automatically imported in the tables because the parameter ROWS is set to Y by default. Indexes are automatically populated because the parameter INDEXES is set to Y by default. Indexes are created along with the table and hence updated along with data import.

Import Process Sequence (continued)

4 Triggers are imported and integrity constraints are enabled on new tables.

The order in which you import tables may be important if you do not import all the objects that a user owns. For, example, if the table with the foreign key has a referential check on the table with the primary key, and the foreign key table is imported first, then all rows that reference the primary key that have not been imported will be rejected if the constraints were enabled. For a full database export this is not a problem.

NLS Considerations

NLS Considerations

- **The export file identifies the character encoding scheme used for the character data in the file.**
- **The import utility translates data to the character set of its host system.**
- **A multibyte character set export file must be imported into a system that has the same characteristics.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

NLS Considerations

When moving data from one Oracle database by using one character set to a database with a different character set, ensure the data conversion must be handled appropriately. You can do this by setting the NLS_LANG environment variable to the character set definition of the database from which the data is being exported. Not setting this correctly could cause unwanted conversion of characters in the data, possibly causing loss of data.

Examples

When converting from a 7-bit ASCII character set, such as American English, to an 8-bit character set such as Danish, no conversion is needed because all characters have an equivalent character in the Danish alphabet.

When converting from an 8-bit ASCII character set, such as Danish, to a 7-bit character set such as American English, causes the extra Danish characters that are not found in the American alphabet to be converted to question marks (?). In this case the question marks are substituted for the unknown Danish characters which is the appropriate result.

Guidelines

- In the 8-bit to 8-bit data movement, whether characters are lost depends upon the specifics of the languages used to enter the data. For example, the Spanish alphabet has letters that are not in the Danish alphabet, so moving data from a Spanish database to a Danish one would result in possible conversion and therefore possible loss of those characters.
- Multibyte to multibyte data movement also depends upon the specifics of the multibyte language.

Tablespace Point-in-Time Recovery (TSPITR)

Tablespace Point-in-Time Recovery

1. Check dependencies and restrictions.
2. Prepare primary database for TSPITR.
3. Create a copy of the database (clone).
4. Recover a subset of the clone database.
5. Copy the cloned data files to the primary db.
6. Export objects definitions from the clone db.
7. Drop objects in the primary database.
8. Import objects definitions into the primary db.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Using TSPITR

This method of recovery can be used if a user error occurs and the database cannot be taken back in time. For example:

- Users notice after a couple of days that an application program has not been working correctly and invalid data exists in a table.
- Taking the database back two days is an impossible request, since there are many users performing different tasks on the database.
- The table must be recovered.

In this case, TSPITR only requires a tablespace and its dependencies to be taken back two days.

Note: This section is designed as an introduction to the TSPITR concept, not a low-level understanding. If backup and recovery procedures are followed, then this method will never be required.

TSPITR Requirements

You must meet the following requirements before using TSPITR:

- All data files that are part of the tablespace requiring recovery must be present.
- You must make a backup of the control file by using the `ALTER DATABASE BACKUP CONTROLFILE TO <NAME>` command.
- There must be enough disk space and memory to start the clone database.

Performing TSPITR

If you require more information on performing tablespace point-in-time recovery, refer to the *Oracle8i: Backup and Recovery Guide*.

The following steps are required for TSPITR:

- 1 Find out if objects will be lost when performing TSPITR.
- 2 Research and resolve dependencies on the primary database.
- 3 Prepare the primary database for TSPITR.
- 4 Prepare the parameter files for the clone database (temporary copy of the database).
- 5 Prepare the clone database for TSPITR.
- 6 Recover a subset of the clone database.
- 7 Open the clone database.
- 8 Prepare the clone database for export.
- 9 Copy the cloned data files to the primary database.
- 10 Export data dictionary metadata about the data file's content from the clone database.
- 11 Import data dictionary metadata about the data file's content into the primary database.
- 12 Prepare the primary database for use.

TSPITR Methods

TSPITR Methods

A TSPITR may be performed in two ways:

- **Traditional O/S method:**
 - Primary and clone databases must reside on the same computer
 - Requires a clone database
- **With transportable tablespace feature:**
 - Does not require primary and target databases to reside on the same computer
 - Does not need a clone database
 - Can recover dropped tablespaces

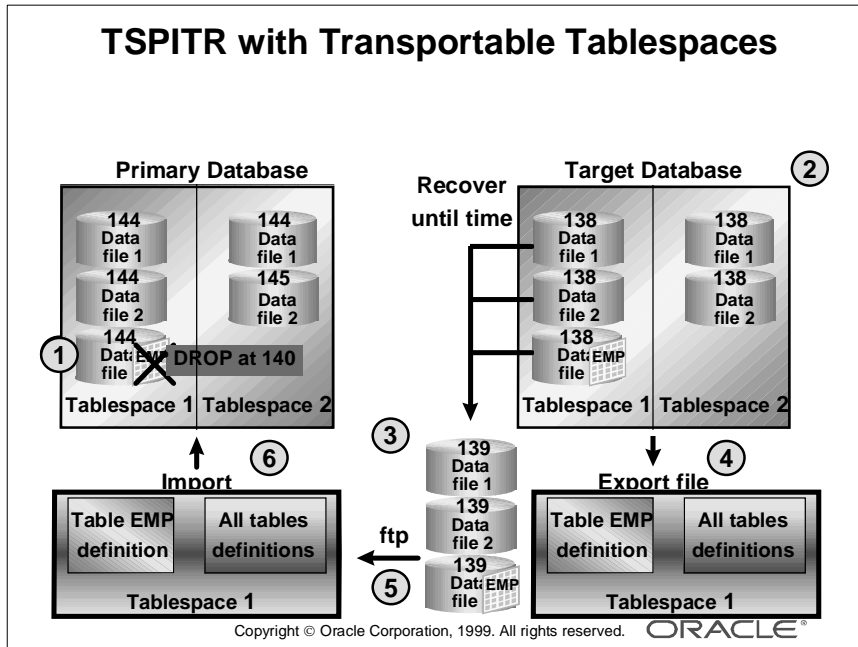
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

The Two Methods

You can perform TSPITR in two different ways:

Method	Consequence
Traditional O/S TSPITR	You must follow special procedures for creating clone <code>init.ora</code> files, mounting the clone database, and so on. O/S TSPITR assumes that the user may place the clone database on the same computer as the primary database. The procedures provide error checks to prevent the corruption of the primary database on the same computer while recovering the clone database.
TSPITR by using the transportable tablespace feature	Performing TSPITR by using transportable tablespaces relaxes this requirement. This method differs from standard O/S TSPITR mainly in using transported tablespaces to perform the last step of TSPITR. You must set the <code>COMPATIBLE</code> initialization parameter to 8.1 or higher to use this method.

TSPITR by Using Transportable Tablespaces



Performing TSPITR by Using Transportable Tablespaces

Note: To learn how to transport tablespaces between databases, see the *Oracle8i Administrator's Guide* or the *Oracle8i Database Administration* course.

- If you restore backups to a different computer separate from the primary database, you can start the target database as if it were the primary database, by using the normal database MOUNT command instead of the clone database MOUNT command.
- If you restore backups on the same computer as the primary database, however, follow the special procedure to create the clone database as described in O/S TSPITR, since this procedure helps prevent accidental corruption of the primary database while recovering the clone database on the same computer.

More Flexibility

TSPITR by way of transportable tablespaces provides basically the same functionality as O/S TSPITR, but is more flexible since:

- You can recover dropped tablespaces, which you cannot do by using O/S TSPITR.
- You can transport the recovered tablespaces from the target database to a different database to test the procedure before making permanent changes to the source database.

Procedure

- 1** Restore the backup to construct the target database. Create the target database on the same computer as the primary database or on a different computer.
- 2** If you create the target database by using the special clone database procedure, place all recovery set and auxiliary set files online:

```
ALTER DATABASE DATAFILE 'datafile_name' ONLINE;
```

If you create the target database as a normal database (on a computer different from the primary database), take all data files not in the recovery and auxiliary set offline:

```
ALTER DATABASE DATAFILE 'datafile_name' OFFLINE;
```
- 3** Recover the target database to the specified point in time.
- 4** Open the target database with the RESETLOGS option.
- 5** Make the tablespaces in the recovery set read-only:

```
ALTER TABLESPACE tablespace_name1 READ ONLY;
```

```
ALTER TABLESPACE tablespace_name2 READ ONLY;
```
- 6** Generate the transportable set by running EXPORT. Include all tablespaces in the recovery set.

```
exp system/manager TRANSPORT_TABLESPACE=y
```

```
TABLESPACES=(tablespace_name1, tablespace_name2)
```
- 7** In the primary database, drop the tablespaces in the recovery set:

```
DROP TABLESPACE tablespace_name1;
```

```
DROP TABLESPACE tablespace_name2;
```
- 8** Copy the data files of the transported tablespaces and the export file to the primary database.
- 9** Plug in the transportable set into the primary database by running IMPORT:

```
imp system/manager TRANSPORT_TABLESPACE=y
```

```
TABLESPACES=(tablespace_name1, tablespace_name2)
```
- 10** If necessary, make the recovered tablespaces read write:

```
ALTER TABLESPACE tablespace_name1 READ WRITE;
```

```
ALTER TABLESPACE tablespace_name2 READ WRITE;
```
- 11** Backup the recovered tablespaces in the primary database:

```
ALTER TABLESPACE tablespace_name1 BEGIN BACKUP;
```

```
!cp datafile_name backup/datafile_name
```

```
ALTER TABLESPACE tablespace_name1 END BACKUP;
```

```
ALTER TABLESPACE tablespace_name2 BEGIN BACKUP;
```

```
!cp datafile_name backup/datafile_name
```

```
ALTER TABLESPACE tablespace_name2 END BACKUP;
```

Summary

Summary

In this lesson, you should have learned how to:

- Use the Export and Import utilities to provide a complete logical backup and recovery strategy
- Explain a logical backup as an alternative way to recover from user error
- Explain the Export utility as a logical backup of data, not a physical backup

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Quick Reference

Context	Reference
Parameters	none
Dynamic performance views	none
Data dictionary views	none
Utilities	exp imp

Additional Recovery Issues

Objectives

Objectives

After completing this lesson, you should be able to do the following:

- **List methods for minimizing downtime**
 - **Parallel recovery**
 - **Start recovering a database with missing data files**
 - **Re-create lost temporary or index tablespaces**
- **Reconstruct lost or damaged control files**
- **List recovery issues associated with read-only tablespaces**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Fast-Start Recovery

Fast-Start Recovery

- **Allows for a fast instance recovery:**
 - **Database is automatically available at the end of the roll-forward phase of recovery.**
 - **Individual user process performs rollback when blocks are requested.**
- **Does not require any DBA intervention**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Instance Recovery Phases

- 1** Unsynchronized files: Oracle determines whether a database needs recovery when unsynchronized files are found. Instance failure, such as a shutdown abort, can cause this to happen. This situation causes loss of uncommitted data, because memory is not written to disk and files are not synchronized before shutdown.
- 2** Roll-forward process: DBWn writes both committed and uncommitted data to the data files. The purpose of the roll-forward process is to apply all changes recorded in the log file to the data blocks.

Note: Rollback segments are populated during the roll-forward phase. Because redo logs store both before and after data images, a rollback segment entry is added if an uncommitted block is found in the data file and no rollback entry exists.

- 3** Committed and uncommitted data in data files: Once the roll-forward phase has successfully been completed, all committed data resides in the data files, though uncommitted data still might exist.

Instance Recovery Phases (continued)

- 4** Rollback phase: To remove the uncommitted data from the files, rollback segments populated during the roll-forward phase are used. Blocks are rolled back when requested by either the Oracle server or a user, depending on who requests the block first.

The database is therefore available even while rollback is yet going on. Only those data blocks participating in rollback are not available.

- 5** Committed data in data files: When both the roll-forward and rollback phases have completed, only committed data resides on disk.
- 6** Synchronized data files: All data files are now synchronized.

Minimize Downtime

Minimize Downtime

- **Parallelize the recovery operation**
- **Bring the database up while recovering the damaged tablespace only**
- **Re-create nonessential damaged tablespaces rather than recovering the files**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Parallel Recovery Operations

Parallel recovery can allow several processes to apply changes from the redo log files. One process, the recovery session, reads the redo log files and dispatches the redo entries to recovery processes. The recovery session dispatches changes to each recovery process. The recovery process applies the change to the appropriate data file.

Starting Oracle with Missing Data Files

If the file loss occurs while the database is down, you will not necessarily know about the loss until you try to start up the system and the startup fails. At this point, you will need to go through the specific steps needed to bring the rest of the database up and available for work, while you go through the recovery operation on the damaged tablespaces.

Following these steps is an excellent way to minimize downtime on your system if just one application tablespace is damaged, work can continue on other applications while you are doing the recovery operation.

Starting Oracle with Missing Data Files (continued)

For example, if the loss was on the Human Resources application file, you will want the rest of the database to be available for work so you can continue to run Manufacturing, General Ledger, Accounts Payable, and Accounts Receivable. The human resources data is unavailable until the recovery is complete, but the unaffected data could still be updated.

Loss of Non Essential Temporary or Index Tablespaces

When a tablespace such as

- A temporary tablespace used for temporary operations
 - An index tablespace used to store indexes that are regularly re-created anyway
- needs recovery, it may take less time to re-create the tablespace rather than recovering it after having restored the associated data files.

Parallel Recovery Operations

Parallel Recovery Operations

- **Parallelism will default to the value set by the RECOVERY_PARALLELISM parameter.**
- **You can specify degree of parallelism by using the PARALLEL keyword.**
- **A value of one or two processes per disk drive containing data files is sufficient for most recovery situations.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

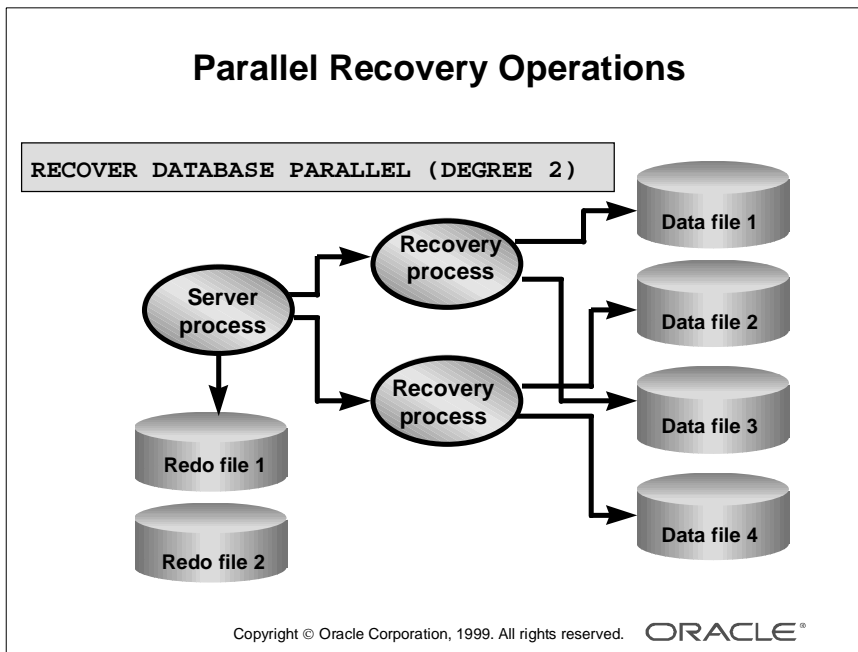
RECOVERY_PARALLELISM Parameter

Specify parallel recovery through initialization parameters or through options on the RECOVER command:

- The value of the parameter specifies the default number of recovery processes per session. If no PARALLEL clause is specified on the RECOVER command, the value of this parameter is used to determine parallelism.
- A value of one or two processes per disk drive containing data files is sufficient for most recovery situations.
- The value is useful for specifying the number of recovery processes to use for instance recovery.

Guidelines

- The number of recovery processes cannot exceed the value of the initialization parameter PARALLEL_MAX_SERVERS.
- A value of zero or one indicates that recovery is to be performed serially by one process.



Parallel Recovery Operations

Use the RECOVER command in SQL Plus Worksheet or SQL*Plus line mode to specify the number of recovery processes to use for media recovery. A value for the PARALLEL clause of the RECOVER command overrides the value of the RECOVERY_PARALLELISM initialization parameter.

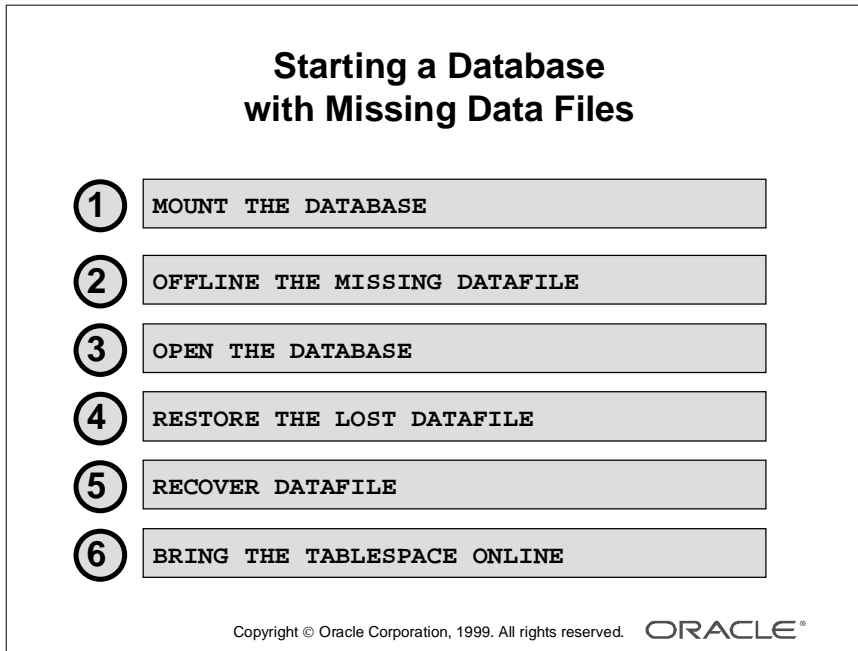
Example

```
SQL> RECOVER DATABASE {PARALLEL ([DEGREE {integer|DEFAULT} |  
INSTANCES {integer|DEFAULT}]...) |NOPARALLEL};
```

where:	PARALLEL	DEGREE specifies the number of recovery processes used to apply redo entries to data files on each instance. DEGREE DEFAULT indicates that twice the number of data files being recovered is the number of recovery processes to use.
	NOPARALLEL	The NOPARALLEL keyword specifies that recovery is to proceed serially.
	DEGREE	The number of recovery processes specified with DEGREE is used on each instance.
	INSTANCES	The total number of recovery processes is the integer specified with DEGREE multiplied by the integer specified with INSTANCES.

Note: This is not the full syntax. You can also specify parallel operations at the TABLESPACE and DATAFILE levels as well as the DATABASE level.

Starting a Database with a Missing Data File



Starting a Database with a Missing Data File

- 1 After receiving the missing data file message at startup, restart the database in the mount mode.

```
SQL> STARTUP MOUNT
```

- 2 Alter the missing data files offline using the ALTER DATABASE DATAFILE *<filename>* OFFLINE command:

```
SQL> ALTER DATABASE DATAFILE '/users/dba00/u01/user01.dbf'
      2 OFFLINE;
```

You may need to use the OFFLINE IMMEDIATE syntax and this requires that your database be running in ARCHIVELOG mode.

- 3 Open the rest of the database for work:

```
SQL> ALTER DATABASE OPEN;
```

- 4 Restore backup copies of the damaged files by using operating system commands.

- 5 Initiate recovery of tablespace or data files, as appropriate by using archived redo logs.

```
SQL> RECOVER TABLESPACE USER_DATA
```

Starting a Database with a Missing Data File (continued)

- 6 Alter the tablespace online again once recover operation is completed.

```
SQL> ALTER TABLESPACE USER_DATA ONLINE;
```

At this point the recovered tablespace will have been rolled forward to the current time, even while the rest of the database continued to be updated during the recovery operation. The recovery operation will recover through the current online redo logs, so the log sequence number in the headers of the recovered files will match the log sequence number in the headers of all the other files in the database. In this way, the system will resynchronize the files at the end of the recovery, with no loss of committed transaction activity.

Loss of Non-Essential Data File

Loss of Non Essential Tablespaces

- ① MOUNT THE DATABASE
- ② OFFLINE DROP THE MISSING DATAFILE
- ③ OPEN THE DATABASE
- ④ DISABLE PRIMARY KEY CONSTRAINTS (INDEX tbs)
- ⑤ DROP THE TEMP or INDEX TABLESPACE
- ⑥ RE-CREATE THE TEMP or INDEX TABLESPACE
- ⑦ REENABLE PRIMARY KEY and FOREIGN KEY CONSTRAINTS (INDEX tbs)

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE**®

Loss of TEMP or INDEX Tablespaces

Rather than restoring the data files associated with the damaged tablespace and then recovering the temporary or index tablespace, re-create the tablespace instead.

Recovery of TEMP or INDEX Tablespaces

- 1 Start the instance if necessary.
- 2 Shut down the instance if the start failed.
- 3 Mount the database.
- 4 Query the V\$RECOVER_FILE and V\$DATAFILE views to note the file names that correspond to the file numbes:

```
SQL> select * from v$recover_file;
```

FILE#	ONLINE	ERROR	CHANGE#	TIME
----	-----	-----	-----	----
4	ONLINE	FILE NOT FOUND	0	
6	ONLINE	FILE NOT FOUND	0	

Recovery of TEMP or INDEX Tablespaces (continued)

- 5**
- Alter the database to take the file offline and drop it:

```
SQL> alter database datafile
      2 'D:\ORANT8I\ORADATA\ORC8\TEMP01.DBF' offline drop;
Database altered.
SQL> alter database datafile
      2 'D:\ORANT8I\ORADATA\ORC8\INDX01.DBF' offline drop;
Database altered.
```

- 6**
- Open the database:

```
SQL> alter database open;
Database altered.
```

- 7**
- Drop the temp tablespace:

```
SQL> select * from v$tablespace;

TS#      NAME
-----
0        SYSTEM
3        TEMP
2        RBS
5        INDX
```

...

```
SQL> drop tablespace temp;
Tablespace dropped.
```

- 8**
- Create the TEMP tablespace by using the same filename noted in Step 5, using a size of 500K and the reuse option:

```
SQL> create tablespace temp datafile
      2 'D:\ORANT8I\ORADATA\ORC8\TEMP01.DBF' size 500k;
Tablespace created.
```

- 9**
- To drop the INDX tablespace, first disable the primary key constraints so that the DROP TABLESPACE CASCADE CONSTRAINTS command can complete:

```
SQL> select pk.OWNER, pk.CONSTRAINT_NAME, pk.constraint_type,
      2      pk.TABLE_NAME, fk.OWNER, fk.CONSTRAINT_NAME,
      3      fk.constraint_type, fk.TABLE_NAME
      4 from dba_constraints pk, dba_constraints fk
      5 where pk.constraint_name = fk.r_constraint_name(+)
      6 and   pk.table_name in (select table_name
      7                          from dba_indexes
      8                          where tablespace_name='INDX')
      9* and   pk.constraint_type = 'P';
```

Recovery of TEMP or INDEX Tablespaces (continued)

OWNER	CONSTRAINT_NAM	C	TABLE_NAME	OWNER	CONSTRAINT_NAM	C	TABLE_NAME
-----	-----	-	-----	-----	-----	-	-----
DEMO	SYS_C001352	P	CUSTOMER	DEMO	SYS_C001370	R	SALES_ORDER
DEMO	SYS_C001353	P	DEPARTMENT	DEMO	SYS_C001366	R	EMPLOYEE
DEMO	SYS_C001354	P	EMPLOYEE	DEMO	SYS_C001362	R	CUSTOMER
DEMO	SYS_C001354	P	EMPLOYEE	DEMO	SYS_C001365	R	EMPLOYEE
DEMO	SYS_C001356	P	JOB	DEMO	SYS_C001364	R	EMPLOYEE
DEMO	SYS_C001357	P	LOCATION	DEMO	SYS_C001363	R	DEPARTMENT
DEMO	SYS_C001359	P	PRODUCT	DEMO	SYS_C001368	R	ITEM
DEMO	SYS_C001359	P	PRODUCT	DEMO	SYS_C001369	R	PRICE
DEMO	SYS_C001361	P	SALES_ORDER	DEMO	SYS_C001367	R	ITEM
DEMO	SYS_C001355	P	ITEM				
DEMO	SYS_C001360	P	SALARY_GRADE				

9 rows selected.

```
SQL> alter table demo.customer disable primary key cascade;
```

Table altered.

```
SQL> alter table demo.department disable primary key cascade;
```

Table altered.

...

```
SQL> drop tablespace indx including contents cascade constraints;
```

Tablespace dropped.

```
SQL> create tablespace indx datafile
```

```
2      'D:\ORANT8I\ORADATA\ORC8\INDX01.DBF' size 500K;
```

Tablespace created.

10 Reenable the primary key and foreign key constraints:

```
SQL> alter table demo.customer enable primary key
```

```
2      using index tablespace indx;
```

Table altered.

```
SQL> alter table demo.department enable primary key
```

```
2      using index tablespace indx;
```

Table altered.

...

```
SQL> alter table demo.customer enable constraint SYS_C001362;
```

Table altered.

```
SQL> alter table demo.employee enable constraint SYS_C001365;
```

Table altered.

...

11 Determine whether a full off-line backup is required and perform one if necessary.

Loss of Control Files

Loss of Control Files

You may need to create control files if:

- **All control files are lost because of a failure**
- **The name of a database needs to be changed**
- **The current settings in the control file need to be changed**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Loss of Control Files

Three situations in which a DBA may be confronted with recovering or re-creating a control file are:

- All control files are lost because of a failure. Technically, a DBA should never be confronted with this problem if the control files are mirrored and spread across multiple physical devices.
- If the DBA needs to change the name of a database. This may be necessary if the database is to become part of a distributed environment and other databases have the same name. It may also be necessary when restoring a database for recovery purposes.
- Settings in the control file are fixed at the time the control file is created. These include MAXDATAFILES, MAXLOGMEMBERS, and so on. Once the control file is created, these settings cannot be dynamically altered and the DBA must re-create the control file.

Recovering Control Files

Recovering Control Files

Methods to recover from loss of control file:

- **Use the current control file**
- **Create a new control file**
- **Use a backup control file**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Recovering Control Files

There are three ways to restore a control file:

- Use a current copy if available. This assumes that you have not lost all of your control files because a multiplexed copy exists.
- Use the `CREATE CONTROLFILE` command to create a new file. To do this you must know all of the files for the database. Backing up a control file to trace on an occasional basis will help facilitate this process.
- Use the `BACKUP CONTROLFILE` command. This will be necessary if all control files have been lost.

Note: If your database is properly configured with mirrored control files, you should never experience loss of all control files.

Example

When performing a recovery operation, you will know when to recover by using a backup control file if you receive the following error message:

```
SQL> alter database open;
ORA-00283: Recovery session canceled due to errors
ORA-01122: database file 1 failed verification check
ORA-01110: data file 1: '/users/dba00/u01/sys01.dbf'
ORA-01207: file is more recent than control file - old control file
```

The steps to recover from this problem include:

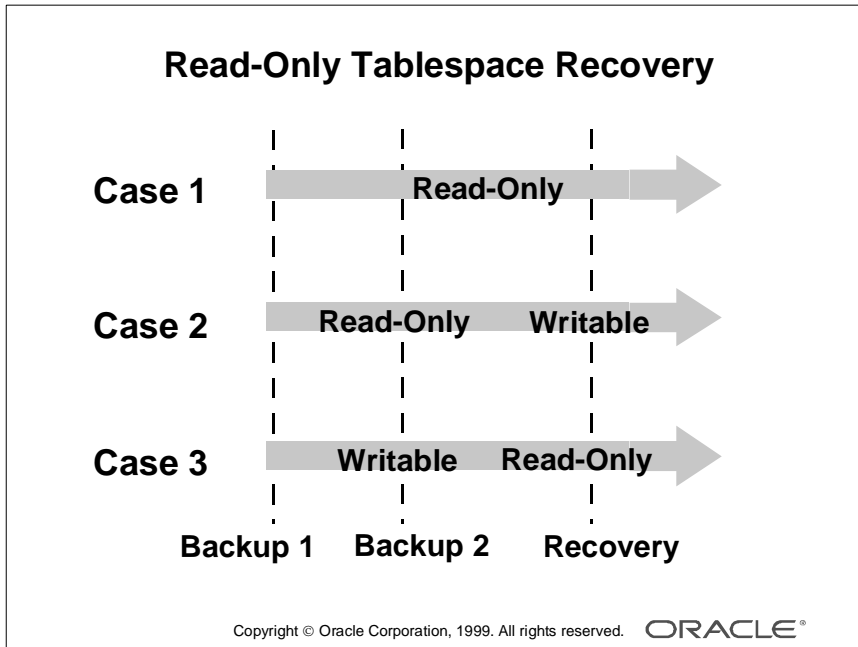
- 1** Restore a backup copy of the control file that was created using the ALTER DATABASE BACKUP CONTROLFILE command.
- 2** Recover the database by using the backup control file:

```
SQL> recover database using backup controlfile;
```

- 3** Open the database by using the RESETLOGS option:

```
SQL> alter database open resetlogs;
```

Read-Only Tablespace Recovery



Recovery Cases

Case 1 The tablespace being recovered is read-only, and was read-only when the last backup occurred. In this case, you can simply restore the tablespace from the backup. There is no need to apply any redo information.

Case 2 The tablespace being recovered is writable, but was read-only when the last backup occurred. In this case, you need to restore the tablespace from the backup and apply the redo information from the point when the tablespace was made writable.

Case 3 The tablespace being recovered is read-only, but was writable when the last backup occurred. Because you should always back up a tablespace after making it read-only, you should not experience this situation. However, if this does occur, you must restore the tablespace from the backup and recover up to the time that the tablespace was made read-only.

Read-Only Tablespace Recovery Issues

Read-Only Tablespace Recovery Issues

Special considerations must be taken for read-only tablespaces when:

- **Re-creating a control file**
- **Renaming data files**
- **Using a backup control file**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Re-creating a Control File

If you need to re-create a control file with the `CREATE CONTROL FILE` command and your database has read-only tablespaces, you must follow special procedures. Issue the command `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` to get a listing of the procedures.

Renaming Data Files

If you cannot restore a copy of the data files in a read-only tablespace to the correct destination, you can use the `ALTER DATABASE RENAME` command to place the files in a new location.

Backup Control File

The procedure for recovering read-only tablespaces with a backup control file is essentially the same as for offline normal tablespaces, except that you need to bring the tablespace online after the database is open.

Summary

Summary

In this lesson, you should have learned how to:

- **Minimize downtime by:**
 - Starting the database and making it available for use even if data files are missing
 - Determining the nature of the tablespaces, essential or non essential
 - Using parallel recovery operations
- **Prevent recoveries by having a good strategy, for example, mirrored control files**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Quick Reference

Context	Reference
Parameters	MAXLOGMEMBERS PARALLEL_MAX_SERVERS RECOVERY_PARALLELISM
Dynamic performance views	V\$RECOVER_FILE V\$DATAFILE
Data dictionary views	None
Commands	ALTER DATABASE DATAFILE [ONLINE OFFLINE] ALTER DATABASE OPEN ALTER DATABASE BACKUP CONTROLFILE TO TRACE ALTER TABLESPACE [ONLINE OFFLINE] CREATE CONTROLFILE RECOVER TABLESPACE RECOVER DATABASE [PARALLEL NOPARALLEL]

Oracle Utilities for Troubleshooting

Objectives

Objectives

After completing this lesson, you should be able to do the following:

- Use log and trace files to diagnose backup and recovery problems
- Detect corruptions by using different methods
- Detect and mark corrupted blocks by using the DBMS_REPAIR package
- Use the DBVERIFY utility
- Use the LogMiner utility to analyze redo log files to recover by undoing changes

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

The Alert Log File

The Alert Log File

- Records informational and error messages for Oracle
- Reflects the ongoing status of the system
- Is continuously written to, with new messages being appended to the end of the alert file
- Shows date and time of startup, shutdown, recovery operations, and so on

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

The Alert Log File

The `alert.log` file is a file written by the Oracle server that includes status information for the instance and database. It is the first place the DBA should look for information in the event of a system problem, unless the problem is obvious. Anything that affects the database instance wide or globally is recorded in the `alert.log`, such as:

- All instance startups and shutdowns
- Every CREATE, ALTER, or DROP operation on a rollback segment, tablespace, or database
- Database errors and events

Because the file is appended to continuously, DBAs need to periodically copy it to an alternate storage device for historical information, then purge it before it fills up the disk.

Example

The following output shows the errors recorded in the `alert.log` file because of a missing online redo log file:

```
...
Tue Nov 25 19:30:25 1997
Errors in file /users/dba00/trace/dba00_lgwr_14838.trc:
ORA-00321: log 2 of thread 1, cannot update log file header
ORA-00312: online log 2 thread 1: '/users/dba00/u02/log2b.rdo'
Tue Nov 25 19:30:25 1997
Errors in file /users/dba00/trace/dba00_lgwr_14838.trc:
ORA-00313: open failed for members of log group 2 of thread 1
Thread 1 advanced to log sequence 1312
Current log# 2 seq# 1312 mem# 0: /users/dba00/u02/log2a.rdo
...
```

Because there were two members in the group, the fact that LGWR was able to continue processing shows the importance of monitoring the contents of the `alert.log` file output on a regular basis to determine if problems exist.

In addition to the `alert.log` file, the DBA can use trace files to help diagnose problems. The example above also caused the LGWR process to create a trace file.

Oracle Trace Files

Oracle Trace Files

- Created by Oracle background processes
- Written to destination specified by the `BACKGROUND_DUMP_DEST` parameter
- Are source of additional diagnostic information

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Oracle Trace Files

If one of the Oracle background processes encounters an error, it usually creates a trace file with information about the error. The name of the background process that generated the trace file such as PMON, SMON, LGWR, DBW n , and so on is included as part of the trace file name.

Trace files are written to the destination as specified by the `BACKGROUND_DUMP_DEST` parameter. The DBA should monitor trace files on a regular basis to help determine database problems. These files also need to be purged periodically.

Example

The following is an output from a `dba00_lgwr_14838.trc` file that was created because of a missing online redo log file.

```
...
ORA-00313: open failed for members of log group 2 of thread 1
ORA-00312: online log 2 thread 1: '/users/dba00/u02/log2b.rdo'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
ORA-00321: log 2 of thread 1, cannot update log file header
ORA-00312: online log 2 thread 1: '/users/dba00/u02/log2b.rdo'
ORA-00313: open failed for members of log group 2 of thread 1
...
```

Note: The number of the trace file, 14838, was also listed in the previous `alert.log` output.

Corrupt Block Detection and Repair

Corrupt Block Detection and Repair

An experienced DBA can:

- Detect and report block corruptions early by using various methods
- Make corrupted objects usable more easily
- Repair some corruptions

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Facilities to Detect and Repair Corrupted Blocks

There are several facilities for detecting and recovering from block corruptions. They are:

- Using the initialization parameters DB_BLOCK_CHECKING and DB_BLOCK_CHECKSUM
- Using the new package DBMS_REPAIR
- Using the SQL command ANALYZE VALIDATE STRUCTURE
- Using the DBVERIFY utility to check files online

Repairing block corruptions requires knowledge of the dependencies between objects and must be attempted with extreme care. Otherwise, the integrity of the data could be compromised. These operations should generally be carried out with help from Oracle Support.

Methods for Detecting Block Corruption

Methods for Detecting Block Corruption	
Method	Description
DB_BLOCK_CHECKSUM	Parameter to check data file and log blocks when blocks are changed
DB_BLOCK_CHECKING	Parameter to check data and index blocks when changes are made
DBVERIFY	Utility to perform online checks
DBMS_REPAIR	Package to detect and report corruptions for a table or index
ANALYZE VALIDATE STRUCTURE	Command to check and report structure integrity for a table or index

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

DB_BLOCK_CHECKSUM Parameter

DBWn and the direct loader calculate a checksum and store it in the cache header of every data block when writing it to disk. Checksums are verified when a block is read. Every log block will also be given a checksum before it is written to the current log.

DB_BLOCK_CHECKING Parameter

The DB_BLOCK_CHECKING parameter can be set to TRUE to check data and index blocks whenever they are changed.

DBVERIFY Utility

This command line utility performs integrity checks on data file blocks. The advantage of this utility is that it has minimal impact on online performance and can be used to verify both online and backup data files.

For a discussion of this utility, refer to the *Oracle8i Utilities* manual.

DBMS_REPAIR Package

The DBMS_REPAIR package provides a new PL/SQL interface to detect and mark block corruptions. This package is created by running the `dbmsrpr.sql` and `prvtrpr.plb` scripts as the user SYS.

ANALYZE TABLE ... VALIDATE STRUCTURE

The ANALYZE TABLE...VALIDATE STRUCTURE statement validates the structure of the analyzed object.

- If the Oracle server successfully validates the structure, a message confirming its validation is returned to you.
- If the Oracle server encounters corruption in the structure of the object, an error message is returned to you. In this case, you would drop and re-create the object.

DB_BLOCK_CHECKSUM Parameter

Detecting Block Corruption Using Checksums

- **DB_BLOCK_CHECKSUM = TRUE**
- **Every data and log block is given a checksum before a write.**
- **Checksum is used to check for corruption when a block is read.**
- **Checksum returns an error and writes information to the trace file.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Checksum Operation

- The `init.ora` parameter, `DB_BLOCK_CHECKSUM`, in the `init.ora` parameter file is set to `TRUE` or `FALSE`.
- If the value of `DB_BLOCK_CHECKSUM` is `TRUE`, a checksum for every block, including temporary blocks, is computed and stored in the block's header. The default value for `DB_BLOCK_CHECKSUM` is `FALSE`.
- The next time the Oracle server reads a data or log block, it uses the checksum to detect corruption in the block. If corruption is detected, the Oracle server returns the message `ORA-01578` and writes information about the corrupted block to a trace file.

Note: Setting `DB_BLOCK_CHECKSUM` to `TRUE` causes performance overhead. Set this value to `TRUE` only under the advice of Oracle Support personnel to diagnose data corruption problems.

DB_BLOCK_CHECKING Parameter

Other Methods of Detecting Block Corruption

New initialization parameter:

```
DB_BLOCK_CHECKING = true
```

- Performs logical block check on data and index blocks when they are changed
- Raises ORA-1578 if corruption is found, and dumps information to trace files

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

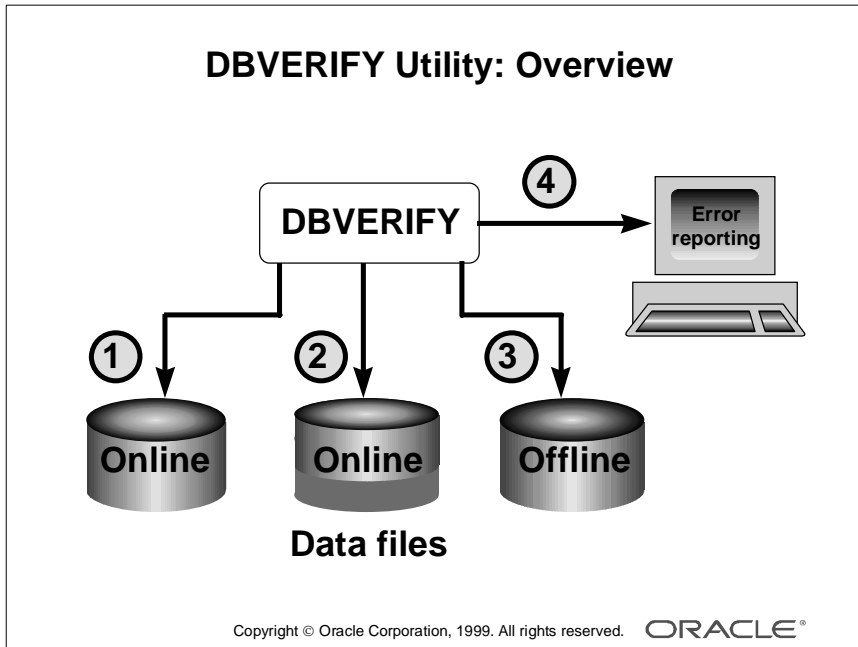
DB_BLOCK_CHECKING

This parameter can be altered dynamically and has minimal performance impact because it does not check blocks while reading. If this parameter is set, the Oracle server generates the following error when it encounters a corrupted block, and writes it to the alert file:

```
ORA-1578 ORACLE data block corrupted
```

The main benefits of this new parameter is that it detects any possible block corruptions immediately, before the effect can become too widespread. This parameter will eventually replace existing events 10210 and 10211.

DBVERIFY Utility



DBVERIFY Utility

The DBVERIFY utility enables administrators to perform verification of data files by checking the structural integrity of data blocks within specified data files. That the utility is external to the database minimizes the impact on database activities.

DBVERIFY Main Features

Step	Explanation
1	The utility can be used to verify online data files.
2	You can invoke the utility on a portion of a data file.
3	The utility can be used to verify online data files.
4	You can direct the output of the utility to an error log.

Running DBVERIFY

The name of the executable for the DBVERIFY utility varies across operating systems. It is located in the bin directory of the appropriate Oracle Home.

The name of the executable is OS-dependent. For UNIX you execute the dbv executable.

The Command Line Interface

Command Line Interface

- External command line utility

```
%dbv file=/users/DBA00/data01.dbf logfile=dbv.log
```

- Used to ensure that a backup database or data file is valid before a restore
- May be a helpful diagnostic aid when data-corruption problems are encountered

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Command Line Interface

You invoke the DBVERIFY utility is invoked by using a command line interface. You use this utility primarily when you need to ensure that a backup database (or data file) is valid before it is restored or as a diagnostic aid when you have encountered data-corruption problems.

Example

To verify the integrity of the `data01.dbf` data file, starting with block 1 and ending with block 500, you execute the following command:

UNIX

```
$ dbv /users/DB00/u03/data01.dbf start=1 end=500
```

DBVERIFY Output

An example of the output from the previous command would look like the following:

```
DBVERIFY - Verification starting : FILE = /users/DBA00/u03/
data01.dbf

DBVERIFY - Verification complete

Total Pages Examined :          500
Total Pages Processed (Data):    22
Total Pages Failing   (Data):    0
Total Pages Processed (Index):   16
Total Pages Failing (Index):     0
Total Pages Empty :             0
Total Pages Marked Corrupt:      0
Total Pages Influx:             0
```

where: Pages is the number of Oracle blocks processed.

DBVERIFY Parameters

Parameter	Description
FILE	Name of database file to verify.
START	Starting block address to verify. Block address is specified in Oracle blocks. If START is not specified it assumes the first block in the file.
END	The ending block address to verify. If END is not specified, it assumes the last block in the file.
BLOCKSIZE	Required only if the file has a block size greater than 2KB.
LOGFILE	Specifies the file to which logging information should be written. Default is to send output to the terminal display.
FEEDBACK	Causes DB_VERIFY to display a single '.' for <i>n</i> pages verified.
HELP	Provides on-screen help.
PARFILE	Specifies the name of the parameter file to use.

DBMS_REPAIR Package

Detecting Block Corruption Using DBMS_REPAIR

- **Create repair table:**

```
dbms_repair.admin_tables('REPAIR_TABLE',  
    DBMS_REPAIR.REPAIR_TABLE,  
    DBMS_REPAIR.CREATE_ACTION,'temp_data');
```

- **Check object for corruptions:**

```
dbms_repair.check_object('ORATRAN',  
    'LOCATIONS',corrupt_count=>:cc);
```

- **Populates repair table**
- **Can check table or index**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Checking Corruption with DBMS_REPAIR

The DBMS_REPAIR package can be used as the user SYS to identify block corruptions in tables, partitions, or indexes. When this feature is used, it checks the objects and generates a list of blocks that are corrupted.

Setting Up for Detecting Block Corruption

Before an object can be checked for block corruption, a table to store the results of this check needs to be created. The procedure DBMS_REPAIR.ADMIN_TABLES can be used to create the table. This procedure accepts the following arguments:

- **TABLE_NAME:** Name of the table to be created for storing data about corrupted blocks.
- **TABLE_TYPE:** REPAIR_TABLE: If a table is checked for corrupt blocks, or ORPHAN_TABLE if an index is checked (discussed later in this section).
- **ACTION:** CREATE_ACTION: Specifies that the table is being created. PURGE_ACTION deletes data that relates to nonexistent objects, while DROP_ACTION drops the table.

Setting Up for Detecting Block Corruption (continued)

- **TABLESPACE:** Name of the tablespace to store the table if the action is **CREATE_ACTION**.

Calling this procedure with **CREATE_ACTION** creates the table and a view with **DBA_** prefix. The example shown creates the **REPAIR_TABLE** table and **DBA_REPAIR_TABLE** view.

Detecting Block Corruption

The procedure **DBMS_REPAIR.CHECK_OBJECT** can be used to check a table, partition, or index for corruptions. This procedure checks all blocks for the specified object and populates the repair table specified with block addresses of the corrupted blocks.

The procedure has the following parameters:

- **SCHEMA_NAME:** Name of the schema that owns the object to be checked
- **OBJECT_NAME:** Name of the object to be checked
- **PARTITION_NAME:** Name of the partition if checking only one partition
- **OBJECT_TYPE:** **TABLE_OBJECT** if checking a table or **INDEX_OBJECT** if checking an index
- **REPAIR_TABLE_NAME:** Name of the table to store information about corrupted blocks
- **FLAGS:** Not used at present
- **RELATIVE_FNO:** Relative file number when checking a block range
- **BLOCK_START:** Starting block number when checking a block range
- **BLOCK_END:** Ending block number when checking a block range
- **CORRUPT_COUNT:** An out variable to store the number of corrupted blocks

Detecting Block Corruption (continued)

After a table has been analyzed, a query such as the following can be executed to check for block corruptions:

```
SQL> SELECT object_name, relative_file_no, block_id,
         2  marked_corrupt, corrupt_description, repair_description
         3  FROM repair_table;
```

OBJECT_NAME	RELATIVE_F	BLOCK_ID	MARKED_COR	CORRUPT_DESCRIPTION	REPAIR_DESCRIPTION
CLASSES	6	3	FALSE	kdbchk: row locked by non-existent transaction table=0 slot=0 lockid=32 ktbhhitc=1 mark block software corrupt	

The repair table indicates that block 3 of file 6 is corrupt, but this block has not yet been marked as corrupt. Therefore, at this time you can extract any rows in the blocks that contain meaningful data. After the block is marked corrupt, you must skip the entire block.

Mark Corrupted Objects

Making Objects Usable

- **Mark blocks as corrupt:**

```
dbms_repair.fix_corrupt_blocks('ORATRIN',
                              'LOCATIONS',fix_count=>:fc);
```

- Uses repair table to identify blocks
- Records time of fix in repair table

- **Enable skipping of corrupt blocks:**

```
dbms_repair.skip_corrupt_blocks(
                              'ORATRIN', 'LOCATIONS');
```

- **Recover available data by using table scan**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Marking Corrupt Blocks

When performing a full table scan on a table with corrupt blocks, the user receives errors that cause the scan to abort. To enable reading of all other data from table, a database administrator needs to repair the blocks as far as possible and then skip blocks that cannot be fixed.

The initial release of Oracle8i does not attempt to fix corrupt blocks. At this time, by running the procedure DBMS_REPAIR.FIX_CORRUPT_BLOCKS, a database administrator is only able to mark the block as software corrupt as seen from executing the following query:

```
SELECT object_name, relative_file_no, block_id, marked_corrupt,
       fix_timestamp
FROM repair_table;
```

OBJECT_NAME	RELATIVE_F	BLOCK_ID	MARKED_COR	FIX_TIMES
-----	-----	-----	-----	-----
CLASSES	6	3	TRUE	07-JAN-98

The columns MARKED_CORRUPT and FIX_TIMESTAMP have been updated in this example.

Skipping Corrupted Blocks

The procedure `DBMS_REPAIR.SKIP_CORRUPT_BLOCKS` is used to enable reading all the blocks except those that are marked corrupt.

The procedure has the following parameters:

- `SCHEMA_NAME`: Owner of the object
- `OBJECT_NAME`: Name of the object
- `OBJECT_TYPE`: `TABLE_OBJECT` if table or `CLUSTER_OBJECT` if index
- `FLAGS`: `SKIP_FLAG` to enable skipping of corrupt blocks or `NOSKIP_FLAG` to disable skipping of corrupt blocks. Setting `NOSKIP_FLAG` generates ORA-1578 errors.

After the procedure is executed, a query such as the following can be executed to check for block corruptions:

```
SELECT table_name, skip_corrupt
FROM dba_tables
WHERE table_name = 'CLASSES';
```

TABLE_NAME	SKIP_COR
-----	-----
T1	ENABLED

Skipping Blocks

Implications of Skipping Blocks

- All rows in blocks marked as corrupt are inaccessible.
- Indexes may point to blocks marked corrupt.
- Referential integrity constraints may be violated:
Disable and reenable constraints to identify violations.
- If the head of a freelist is corrupt, the freelist is reinitialized and an error is returned. Repair freelist by using REBUILD_FREELISTS.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Indexes and Corrupt Tables

Checking Index Entries Pointing to Rows in Corrupt Data Blocks

- Create table to hold results:

```
dbms_repair.admin_tables('ORPHAN_TAB1',  
  DBMS_REPAIR.ORPHAN_TABLE,  
  DBMS_REPAIR.CREATE_ACTION,'temp_data');
```

- Check index:

```
dbms_repair.dump_orphan_keys('ORATRIN',  
  'LOC_PK', orphan_table_name=>'ORPHAN_TAB1',  
  key_count=>:kc);
```

Populates orphan table

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Orphan Keys and Inconsistent Results

When skipping of corrupt blocks is enabled on a table, you may find that queries that only probe the index and those that read the table may produce inconsistent results. The DBMS_REPAIR package provides facilities to identify indexes that point to blocks that are marked corrupt in a table. Perform the following steps to identify potential problems with inconsistent indexes and rebuild them:

- Create the table to hold results on checking the index, as shown in the first code box.
- Check the indexes for the table for orphan keys by using the DBMS_REPAIR.DUMP_ORPHAN_KEYS procedure as shown. Orphan keys are index entries that point to blocks in the table marked corrupt. The procedure DUMP_ORPHAN_KEYS uses the following arguments:
 - SCHEMA_NAME: Owner of the index
 - OBJECT_NAME: Index name
 - PARTITION_NAME: Name of the index partition, if analyzing a partition
 - OBJECT_TYPE: Only INDEX_OBJECT permitted currently

Orphan Keys and Inconsistent Results (continued)

- REPAIR_TABLE_NAME: Name of the repair table containing the list of corrupted blocks for the table
- ORPHAN_TABLE_NAME: Name of the orphan table to store the results of the check
- FLAGS: Not used currently
- KEY_COUNT: Output variable containing the number of keys identified

This procedure populates the named orphan table with data about index keys that point to the corrupt blocks.

- From the orphan table, identify the indexes that are likely to cause problems. The following is a sample query that you can use:

```
SELECT index_name, count(*)
FROM orphan_key_table
WHERE table_name = 'CLASSES'
GROUP BY index_name;
```

INDEX_NAME	COUNT (*)
-----	-----
CLASSES_PK	3

- Rebuild indexes that have orphan keys.

Limitations of DBMS_REPAIR

Limitations of DBMS_REPAIR

- Tables with LOBs, nested tables, and VARRAYs can be analyzed, but out-of-line columns are ignored.
- Index-organized tables and LOB indexes cannot be analyzed.
- DUMP_ORPHAN_KEYS does not operate on bitmap and function-based indexes.
- DUMP_ORPHAN_KEYS processes only keys up to 3,950 bytes long.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

ANALYZE VALIDATE STRUCTURE Command

Other Methods of Detecting Block Corruption

ANALYZE VALIDATE STRUCTURE command can be used to check tables and indexes structural integrity and consistency.

- The following statement validates the structure of the index **PARTS_INDEX**:

```
ANALYZE INDEX parts_index VALIDATE STRUCTURE;
```

- The following statement analyzes the **EMP** table and all of its indexes:

```
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Objects Analyzed

The command validates the structure of the analyzed object:

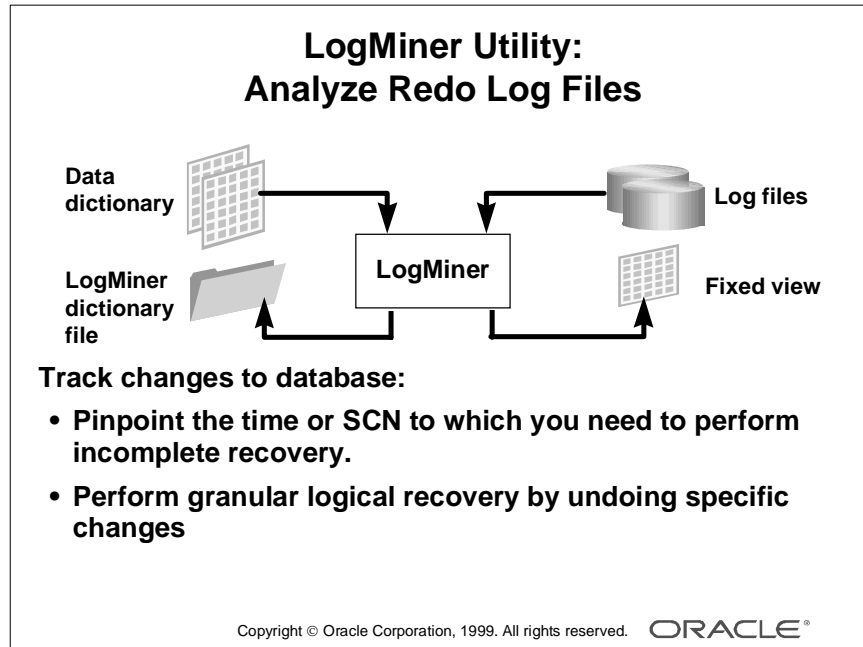
- For a table, Oracle verifies the integrity of each of the table's data blocks and rows.
- For a cluster, Oracle automatically validates the structure of the cluster's tables.
- For a partitioned table, Oracle also verifies that the row belongs to the correct partition. If the row does not collate correctly, the rowid is inserted into the **INVALID_ROWS** table.
- For a temporary table, Oracle validates the structure of the table and its indexes during the current session.
- For an index, Oracle verifies the integrity of each data block in the index and checks for block corruption. This clause does not confirm that each row in the table has an index entry or that each index entry points to a row in the table. You can perform these operations by validating the structure of the table with the **CASCADE** clause.

INTO and CASCADE Clauses

- **INTO** specifies a table into which Oracle lists the rowids of the partitions whose rows do not collate correctly. If you omit schema, Oracle assumes the list is in your own schema. If you omit this clause altogether, Oracle assumes that the table is named `INVALID_ROWS`. The SQL script used to create this table is `utlvalid.sql`.
- **CASCADE** validates the structure of the indexes associated with the table or cluster. If you use this clause when validating a table, Oracle also validates the table's indexes. If you use this clause when validating a cluster, Oracle also validates all the clustered tables' indexes, including the cluster index.

If you use this clause to validate an enabled (but previously disabled) function-based index, validation errors may result. In this case, you must rebuild the index.

LogMiner Utility



Usage

- LogMiner is useful for identifying and undoing logical corruption. It enables you to:
 - Determine when a logical corruption to the database may have begun, pinpointing the time or system change number to which you need to perform incomplete recovery.
 - Perform granular logical recovery by undoing specific changes made by one or more transactions. This minimizes the need for performing point-in-time recovery to recover from a logical application error.
- LogMiner processes redo log files, translating their contents into SQL statements that represent the logical operations performed to the database.
- The V\$LOGMNR_CONTENTS view then lists the reconstructed SQL statements that represent the original operations (SQL_REDO column) and the corresponding SQL statement to undo the operations (SQL_UNDO column).
- Apply the SQL_UNDO statements to roll back the original changes to the database.

Usage (continued)

- LogMiner provides a procedure to analyze and present the data in the form of a virtual table.
- Because the redo log files contain only object identifiers and not object names, to enable interpretation of the object identifiers, the data dictionary information must be available at the time the log files are analyzed.
- LogMiner contains a procedure to extract the data dictionary information into an operating system text file, referred to as the LogMiner dictionary file.
- During the analysis, this information is used along with the records from the log file to populate a virtual table that can be queried using standard SQL.

LogMiner Analysis

Viewing Log Information: Build LogMiner Dictionary File

- Source can be an Oracle8 database:
 - Same platform as the analyzing instance
 - Same character set
- File is used to resolve object names. Perform this step when new objects are added.
- Directory must be specified using the `UTL_FILE_DIR` `init.ora` parameter.
- Execute the package and procedure:

```
dbms_logmnr_d.build(  
  'orcldict.ora', '/oracle/database' );
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

How to Analyze Redo Log File Information

- 1 The first step in analyzing log files is to create the dictionary file.
 - Initialize the `UTL_FILE_DIR` `init.ora` parameter to the directory where the Oracle server will create the dictionary file.
 - Use the `BUILD` procedure in `DBMS_LOGMNR_D` package to generate this file.

Note: If analyzing an Oracle8 database, execute the `dbmslogmnrd.sql` script provided as user `SYS`, to create the `DBMS_LOGMNR_D` package.

The code shown in the example creates the file `/oracle/database/orcldict.ora` and stores the information from the data dictionary. Because the file is used to resolve object names, the data dictionary that is dumped must contain all the objects found in the log file.

Viewing Log Information: Specify Log Files to Be Analyzed

You can specify online or archived files.

1. Create a new list and specify first file:

```
dbms_logmnr.add_logfile('/oracle/database/ORC1/  
log1orcl.ora', dbms_logmnr.NEW);
```

2. Specify additional files to be analyzed:

```
dbms_logmnr.add_logfile('/oracle/database/ORC1/  
log2orcl.ora', dbms_logmnr.ADDFILE);
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

How to Analyze Redo Log File Information (continued)

2 LogMiner can analyze both online and archived log files. Before the analysis can begin, a list of the files to be analyzed is supplied using the DBMS_LOGMNR.ADD_LOGFILE procedure.

- To disregard any file names that have been supplied before, and to create a new list of files to be analyzed, supply the NEW argument while naming the first file in the list.
- While adding other files to the list already created, use the ADDFILE argument as shown in the second example.

The dynamic performance tables must be available when this list is defined.

Viewing Log Information: Analyze Files Specified

- **Initiate log analysis:**

```
dbms_logmnr.start_logmnr(dictfilename=>'/oracle/database  
/orcldict.ora',  
starttime=>to_date('01/01/98:08AM','DD/MM/YY:HHAM'),  
endtime=>to_date('03/01/1998:09AM','DD/MM/YYYY:HHAM'));
```

- Extracts details of transactions that occurred between the times specified
- Can also use system change number ranges.

- **Release resources:**

```
dbms_logmnr.end_logmnr;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

How to Analyze Redo Log File Information (continued)

- 3 Once a list of logs files to be analyzed has been created, you can initiate the analysis by invoking the START_LOGMNR procedure. The parameters that this procedure accepts are:
 - STARTSCN: Lower bound of the SCN range to be analyzed
 - ENDSCN: Upper bound of the SCN to be analyzed
 - STARTTIME: A date value indicating the lower bound of the time range for analysis
 - ENDTIME: A date value indicating the upper bound of the time range for analysis
 - DICTFILENAME: Name of the LogMiner dictionary file
- 4 Use the END_LOGMNR procedure to release resources used by the analyze operation.

Viewing Log Information: Track Database Changes

V\$LOGMNR_CONTENTS.SQL_REDO column
contains statements reflecting changes:

```
SELECT timestamp, username, sql_redo
FROM v$logmnr_contents
WHERE seg_name = 'EMP';

TIMESTAMP USER  SQL_REDO
-----
01-JAN-98  SCOTT  delete from EMP where rowid =
01-JAN-98  SCOTT  insert into EMP(...) ...
02-JAN-98  SCOTT  update EMP set SAL = ... where ...
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

How to Analyze Redo Log File Information (continued)

- 5 The view V\$LOGMNR_CONTENTS can be queried by the session that performed the analysis to view the log information. Other sessions cannot see the data by querying this view. If the results are to be viewed by other sessions or if multiple passes need to be made of the contents of the view, you should store the information from the view into a table.

A partial description of the V\$LOGMNR_CONTENTS view is given below:

Column	Description
SCN	System change number (SCN)
TIMESTAMP	Timestamp
THREAD#	Thread number
LOG_ID	Log ID
XIDUSN	Transaction ID undo segment number
XIDSLOT	Transaction ID slot number
XIDSQN	Transaction ID sequence number
ABS_FILE#	Data block absolute file number
REL_FILE#	Data block relative file number
DATA_BLK#	Data block number
DATA_OBJ#	Data block object number
DATA_OBJD#	Data block data object number

Column	Description
SEG_OWNER	Owner name of segment
SEG_NAME	Segment name
SEG_TYPE_NAME	Type of segment
TABLE_SPACE	Tablespace name of segment
ROW_ID	Row ID
SESSION #	Session number
SERIAL #	Serial number
USERNAME	Username
ROLLBACK	Rollback request
OPERATION	Operation
SQL_REDO	SQL redo
SQL_UNDO	SQL undo
INFO	Informational message
STATUS	Status

How to Analyze Redo Log File Information (continued)

Note

- The SCN provides the appropriate information for performing a point-in-time recovery such as a RECOVER UNTIL SCN.
- The TIMESTAMP provides the exact information for performing a point-in-time recovery such as a RECOVER UNTIL TIME.
- SEG_OWNER and SEG_NAME identify the object on which DML operation occurred.
- USERNAME identifies which user executed the statement.
- SQL_REDO repeats the statement that was executed.
- SQL_UNDO explains how to undo the operation performed in the statement specified in SQL_REDO.

Viewing Log Information: Perform Logical Recovery

The `SQL_UNDO` column shows statements that can be used to reverse the changes:

```
SELECT sql_redo, sql_undo
FROM v$logmnr_contents
WHERE seg_name = 'EMP';
```

SQL REDO	SQL UNDO
-----	-----
delete from EMP ...	insert into EMP(...) ...
insert into EMP ...	delete from EMP ...
update EMP set SAL = ...	update EMP set SAL = ...

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Reverse Changes

The possibility of reversing changes may minimize the need for performing point-in-time recovery to recover from a logical application error if no major dependencies exist on the tables where undo changes are applied.

Views

Obtaining Information About Logs Analyzed

- **V\$LOGMNR_DICTIONARY**
 - **TIMESTAMP**
 - **FILENAME**
- **V\$LOGMNR_LOGS**
 - **LOG_ID, FILENAME**
 - **LOW_SCN, HIGH_SCN**
 - **LOW_TIME, HIGH_TIME**
- **V\$LOGMNR_PARAMETERS**
Arguments supplied during analysis

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Dynamic Views

The dynamic performance views shown can be queried by the analyzing session to verify the dictionary files, log files, and arguments supplied to the analysis.

Limitations

Initial Release of LogMiner: Features and Limitations

- Records are visible only to the analyzing session.
- There is one row per redo record.
- DML on scalar data and transaction control statements are supported.
- DDL statements are shown as DML on data dictionary.
- SQL on chained data rows is not reconstructed.
- Hex values for segment names are shown if:
 - Object definition is not in dictionary file
 - Changes are to clustered tables

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Summary

Summary

In this lesson, you should have learned how to:

- **Locate error messages in the alert log and trace files**
- **Use different tools to detect corrupt blocks**
- **Use the DBMS_REPAIR package to fix corruption and make objects usable**
- **Use the LogMiner utility to:**
 - **Perform a logical recovery instead of an incomplete recovery**
 - **Pinpoint the most exact time for an incomplete recovery**

Copyright ©Oracle Corporation, 1999. All rights reserved. ORACLE®

Quick Reference

Context	Reference
Parameters	BACKGROUND_DUMP_DEST DB_BLOCK_CHECKSUM DB_BLOCK_CHECKING UTL_FILE_DIR
Dynamic performance views	V\$LOGMNR_CONTENTS V\$LOGMNR_DICTIONARY V\$LOGMNR_LOGS V\$LOGMNR_PARAMETERS
Data dictionary views	None
Packages and Procedures	DBMS_REPAIR.admin_tables DBMS_REPAIR.check_object DBMS_REPAIR.fix_corrupt_blocks DBMS_REPAIR.skip_corrupt_blocks DBMS_REPAIR.rebuild_freelists DBMS_REPAIR.dump_orphan_keys DBMS_LOGMNR_D.build DBMS_LOGMNR.add_logfile DBMS_LOGMNR.start_logmnr DBMS_LOGMNR.end_logmnr
Commands	ANALYZE INDEX ... VALIDATE STRUCTURE ANALYZE TABLE... VALIDATE STRUCTURE [CASCADE]
Utility	dbv

