



Oracle Database 19c: Backup and Recovery

Student Guide

D106548GC10

Authors

Donna Keesling

Sean Kim

Technical Contributors and Reviewers

Mark Fuller

James Spiller

Jim Womack

Publisher

Pavithran Adka

1010302020

Copyright © 2019, 2020, Oracle and/or its affiliates.

Disclaimer

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

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

Restricted Rights Notice

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

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Trademark Notice

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

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

Third-Party Content, Products, and Services Disclaimer

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

Contents

1 Backup and Recovery: Overview

- Objectives 1-2
- DBA Responsibilities 1-3
- Separation of DBA Duties 1-4
- Assessing Your Recovery Requirements 1-5
- Categories of Failure 1-7
 - Statement Failure 1-8
 - User Process Failure 1-9
 - Network Failure 1-10
 - User Error 1-11
 - Instance Failure 1-12
 - Media Failure 1-13
 - Data Failures 1-14
 - Instance Recovery 1-15
 - The Checkpoint (CKPT) Process 1-16
 - Redo Log Files and the Log Writer (LGWR) Process 1-17
 - Database Log Mode 1-18
 - Automatic Instance Recovery or Crash Recovery 1-19
 - Phases of Instance Recovery 1-20
 - Tuning Instance Recovery 1-21
 - Using the MTTR Advisor 1-22
 - Restoring and Recovering 1-23
 - Comparing Complete and Incomplete Recovery 1-24
 - The Complete Recovery Process 1-25
 - The Point-in-Time Recovery Process 1-26
 - Oracle Data Protection Solutions 1-28
 - Flashback Technology 1-29
 - Summary 1-30

2 Backup and Recovery Configuration

- Objectives 2-2
- Configuring for Recoverability 2-3
- Configuring the Fast Recovery Area 2-4
- Monitoring the Fast Recovery Area 2-5
- Multiplexing Control Files 2-6

Redo Log Files	2-8
Multiplexing the Redo Log	2-9
Creating Archived Redo Log Files	2-10
Archiver (ARCn) Process	2-11
Archived Redo Log Files: Naming and Destinations	2-12
Configuring ARCHIVELOG Mode	2-13
Summary	2-14
Practice Overview	2-15

3 Using Recovery Manager (RMAN)

Objectives	3-2
Integrated Oracle Recovery Manager (RMAN)	3-3
Connecting to RMAN and a Target Database	3-4
Using SQL in RMAN	3-5
Types of RMAN Commands	3-6
Job Commands: Example	3-7
Configuring Persistent Settings for RMAN	3-8
Viewing Persistent Settings	3-9
Managing Persistent Settings	3-10
Specifying a Retention Policy	3-11
Recovery Window Retention Policy: Example	3-13
Summary	3-14
Practice Overview	3-15

4 Backup Strategies

Objectives	4-2
Understanding Types of Backups	4-3
Backup Terminology	4-4
Understanding Types of Backups	4-5
RMAN Backup Types	4-6
Backup Solutions: Overview	4-8
Balancing Backup and Restore Requirements	4-9
Comparing Backup Strategies	4-11
Option 1: Full and Incremental Backups	4-12
Option 2: Incrementally Updated Disk Backups	4-13
Option 3: Offloading Backups to Physical Standby Database in Data Guard Environment	4-14
Backing Up Read-Only Tablespaces: Considerations	4-15
Data Warehouse Backup and Recovery: Best Practices	4-16
Summary	4-17

5 Creating Database Backups

- Objectives 5-2
- Using RMAN Commands to Create Backups 5-3
- Syntax and Clauses in RMAN 5-4
- Creating Backup Sets 5-5
- Creating Image Copies 5-6
- Creating a Whole Database Backup 5-7
- CDB Backup: Whole CDB Backup 5-8
- CDB Backup: Partial CDB Backup 5-9
- PDB Backup: Partial PDB Backup 5-10
- Review: RMAN Backup Types 5-11
- Incrementally Updated Backups 5-13
- Incrementally Updated Backups: Example 5-14
- Fast Incremental Backup 5-15
- Maintaining the Block Change Tracking File 5-16
- Monitoring Block Change Tracking 5-17
- Automatic Disk-to-Disk Backup and Recovery 5-18
- Oracle-Suggested Backup 5-19
- Backing Up the Control File to a Trace File 5-20
- Cataloging Additional Backup Files 5-22
- Reporting on Backups 5-23
- Using Dynamic Views 5-24
- Summary 5-25
- Practice Overview 5-26

6 Using Optional Backup Features

- Objectives 6-2
- Saving Backup Space with Unused Block Compression 6-3
- Compressing Backups 6-4
- Using RMAN Backup Compression 6-5
- Using a Media Manager 6-6
- Configuring Backup and Restore for Very Large Files 6-8
- Backing Up and Restoring Very Large Files 6-9
- Creating RMAN Multisection Backups 6-10
- Creating Proxy Copies 6-11
- Creating Duplexed Backup Sets by Using BACKUP COPIES 6-12
- Creating Backups of Backup Sets 6-13
- Archival Backups: Concepts 6-14
- Creating Archival Backups with RMAN 6-16
- Managing Archival Database Backups 6-17

Backing Up Recovery Files 6-18

Summary 6-19

Practice Overview 6-20

7 Tuning RMAN Backup Performance

Objectives 7-2

Is There a Problem? 7-3

Diagnosing Performance Bottlenecks 7-4

Diagnosing Performance Bottlenecks: Read Phase 7-5

Is There a "Write" Problem? 7-6

Diagnosing Performance Bottlenecks: Write or Copy Phase 7-7

Using Dynamic Views to Diagnose RMAN Performance 7-8

Monitoring RMAN Job Progress 7-9

Identifying Backup and Restore Bottlenecks 7-11

Asynchronous I/O Bottlenecks 7-12

Synchronous I/O Bottlenecks 7-13

Tuning RMAN Backup Performance 7-14

Parallelization of Backup Sets 7-15

RMAN Multiplexing 7-17

Summary 7-19

Practice Overview 7-20

8 Recovery Catalog Overview

Objectives 8-2

RMAN Repository Data Storage: Comparison of Options 8-3

Storing Information in the Recovery Catalog 8-4

Reasons to Use a Recovery Catalog 8-5

Summary 8-6

9 Creating a Recovery Catalog

Objectives 9-2

Creating the Recovery Catalog: Three Steps 9-3

1. Configuring the Recovery Catalog Database 9-4

2. Creating the Recovery Catalog Owner 9-5

3. Creating the Recovery Catalog 9-6

Summary 9-7

Practice Overview 9-8

10 Managing Target Database Records

Objectives 10-2

Managing Target Database Records in the Recovery Catalog 10-3

Registering a Database in the Recovery Catalog	10-4
Unregistering a Target Database from the Recovery Catalog	10-5
Recovery Catalog Resynchronization: Concepts	10-6
Manually Resynchronizing the Recovery Catalog	10-8
Summary	10-9
Practice Overview	10-10

11 Using Stored Scripts

Objectives	11-2
Using RMAN Stored Scripts	11-3
Executing RMAN Stored Scripts	11-4
Maintaining RMAN Stored Scripts	11-5
Summary	11-6
Practice Overview	11-7

12 Creating and Using Virtual Private Catalogs

Objectives	12-2
Creating and Using Virtual Private Catalogs	12-3
Creating a Virtual Private Catalog	12-4
Managing Virtual Private Catalogs	12-6
Upgrading Virtual Private Catalogs	12-7
Summary	12-8
Practice Overview	12-9

13 Restore and Recovery Concepts

Objectives	13-2
File Loss	13-3
Data Repair Techniques	13-4
Restoring and Recovering	13-5
Using RMAN RESTORE and RECOVER Commands	13-6
Instance Failure	13-7
Instance Recovery	13-8
Phases of Instance Recovery	13-9
Media Failure	13-10
Comparing Complete and Incomplete Recovery	13-11
Complete Recovery Process	13-12
Point-in-Time Recovery Process	13-13
Recovery with the RESETLOGS Option	13-15
Restore and Recovery Performance: Best Practices	13-16
Summary	13-17

14 Diagnosing Failures

Objectives	14-2
Reducing Problem Diagnosis Time	14-3
Automatic Diagnostic Workflow	14-4
Automatic Diagnostic Repository	14-5
ADR Command-Line Tool (ADRCI)	14-6
V\$DIAG_INFO View	14-7
Data Recovery Advisor	14-8
Data Failure: Examples	14-11
Data Recovery Advisor RMAN Command-Line Interface	14-12
Listing Data Failures	14-13
Advising on Repair	14-15
Executing Repairs	14-16
Classifying (and Closing) Failures	14-17
Data Recovery Advisor Views	14-18
Summary	14-19
Practice Overview	14-20

15 Performing Complete Recovery

Objectives	15-2
Ensuring Backups Are Available	15-3
Restoring in NOARCHIVELOG Mode	15-4
Recovery with Incremental Backups in NOARCHIVELOG Mode	15-5
Performing Complete Recovery	15-6
Review: Recovering Image Copies	15-8
Recovering Image Copies: Example	15-9
Performing a Fast Switch to Image Copies	15-10
Using SET NEWNAME for Switching Files	15-11
Using Restore Points	15-12
PDB Tempfile Recovery	15-13
PDB SYSTEM or UNDO Tablespace Recovery	15-14
PDB Non-SYSTEM Tablespace Recovery	15-15
Summary	15-16
Practice Overview	15-17

16 Performing Point-in-Time Recovery

Objectives	16-2
Point-in-Time Recovery	16-3
PITR Terminology	16-4
Performing Point-in-Time Recovery	16-5

When to Use TSPITR	16-7
Tablespace Point-in-Time Recovery: Architecture	16-8
Preparing for TSPITR	16-10
Determining the Correct Target Time	16-11
Determining the Tablespaces for the Recovery Set	16-12
Identifying Objects That Will Be Lost	16-13
Performing RMAN TSPITR	16-14
Performing Fully Automated TSPITR	16-15
Improving TSPITR Performance	16-16
Performing RMAN TSPITR with an RMAN-Managed Auxiliary Instance	16-17
Performing RMAN TSPITR by Using Your Own Auxiliary Instance	16-18
Troubleshooting RMAN TSPITR	16-19
PITR of PDBs	16-20
Recovering Tables from Backups	16-21
Table Recovery: Graphical Overview	16-22
Prerequisites and Limitations	16-23
Specifying the Recovery Point in Time	16-24
Process Steps of Table Recovery	16-25
Summary	16-27
Practice Overview	16-28

17 Performing Block Media Recovery

Objectives	17-2
What Is Block Corruption?	17-3
Block Corruption Symptoms: ORA-01578	17-4
How to Handle Corruption	17-5
Setting Parameters to Detect Corruption	17-6
Block Media Recovery	17-7
Prerequisites for Block Media Recovery	17-8
Recovering Individual Blocks	17-9
Best Practice: Proactive Checks	17-10
Checking for Block Corruption	17-11
Summary	17-12
Practice Overview	17-13

18 Performing Additional Recovery Operations

Objectives	18-2
Recovery from Loss of Server Parameter File	18-3
Restoring the Server Parameter File from the Control File Autobackup	18-4
Loss of a Control File	18-5
Recovering from the Loss of All Control File Copies: Overview	18-6

Restoring the Control File from Autobackup	18-7
Restoring the SPFILE and the Control File	18-8
Recovering NOLOGGING Database Objects	18-9
Loss of a Redo Log File	18-10
Log Group Status: Review	18-11
Recovering from the Loss of a Redo Log Group	18-12
Clearing a Log File	18-13
Re-creating a Password Authentication File	18-14
Summary	18-16
Practice Overview	18-17

19 Oracle Flashback Technology: Overview

Objectives	19-2
Flashback Technologies Error Detection and Correction	19-3
Review: Transactions and Undo	19-4
Flashback Technology	19-5
Summary	19-7

20 Using Logical Flashback Features

Objectives	20-2
Using Flashback Technology to Query Data	20-3
Flashback Query	20-4
Flashback Version Query	20-5
Flashback Table: Overview	20-6
Flashback Table	20-7
Flashback Table: Considerations	20-8
Flashback Transaction Query	20-9
Flashback Transaction Query: Considerations	20-10
Flashback Transaction Backout	20-11
Flashing Back a Transaction	20-12
Best Practices: Undo-Based Flashback Query, Flashback Table	20-13
Flashback Drop and the Recycle Bin	20-14
Recycle Bin	20-15
Bypassing the Recycle Bin	20-16
Using Flashback Data Archives	20-17
Creating a Temporal History and Enabling Archiving	20-18
How the Flashback Data Archive Works	20-19
Collecting User Context in Temporal History	20-20
Transparent Schema Evolution	20-21
Full Schema Evolution	20-22
Temporal Validity and History	20-23

Using the PERIOD FOR Clause	20-24
Filtering on Valid-Time Columns: Example 1	20-25
Filtering on Valid-Time Columns: Example 2	20-26
Using DBMS_FLASHBACK_ARCHIVE	20-27
Summary	20-28
Practice Overview	20-29

21 Using Flashback Database

Objectives	21-2
Preparing Your Database for Flashback	21-3
Guaranteeing Undo Retention	21-4
Flashback Database: Continuous Data Protection	21-5
Flashback Database	21-6
Flashback Database Architecture	21-7
Configuring Flashback Database	21-8
Flashback Database: Examples	21-9
CDB and PDB Flashback	21-10
Flashback Database Considerations	21-11
Monitoring Flashback Database Information	21-12
Guaranteed Restore Points	21-14
Flashback Database and Guaranteed Restore Points	21-15
PDB Flashback and Clean Restore Point	21-16
Best Practices: Flashback Database	21-17
Summary	21-19
Practice Overview	21-20

22 Using PDB Snapshots

Objectives	22-2
PDB Snapshot Carousel	22-3
Creating PDB Snapshots	22-4
Creating PDBs by Using PDB Snapshots	22-5
Dropping PDB Snapshots	22-6
Flashing Back PDBs by Using PDB Snapshots	22-7
Summary	22-8

23 Database Duplication Overview

Objectives	23-2
Using a Duplicate Database	23-3
Choosing Database Duplication Techniques	23-4
Duplicating an Active Database with “Push”	23-5
Comparing the “Push” and “Pull” Methods of Duplication	23-6

Duplicating a Database with a Target Connection	23-7
Duplicating a Database with a Recovery Catalog	23-8
Duplicating a Database Without a Recovery Catalog or Target Connection	23-9
Summary	23-10

24 Creating a Backup-Based Duplicate Database

Objectives	24-2
Creating a Backup-Based Duplicate Database	24-3
Creating an Initialization Parameter File for the Auxiliary Instance	24-4
Specifying New Names for Your Destination	24-5
Using the SET NEWNAME Clauses	24-6
Substitution Variables for SET NEWNAME	24-7
Specifying Parameters for File Naming	24-8
Starting the Instance in NOMOUNT Mode	24-9
Ensuring That Backups and Archived Redo Log Files Are Available	24-10
Allocating Auxiliary Channels	24-11
Understanding the RMAN Duplication Operation	24-12
Specifying Options for the DUPLICATE Command	24-14
Using Additional DUPLICATE Command Options	24-15
Duplicating Selected PDBs in a CDB	24-16
Cloning an Active PDB into an Existing CDB	24-17
Example: Duplicating PDB1 from CDB1 to CDB2 as PDB1	24-18
Example: Duplicating PDB1 from CDB1 to CDB2 as PDB2	24-19
Summary	24-20
Practice Overview	24-21

Backup and Recovery: Overview

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- Identify the responsibilities of a DBA in database backup and recovery
- Identify the types of failure that can occur in an Oracle database
- Describe instance recovery
- Describe complete and incomplete recovery



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

DBA Responsibilities

- Protect the database from failure wherever possible.
- Increase the mean time between failures (MTBF).
- Protect critical components by using redundancy.
- Decrease the mean time to recover (MTTR).
- Minimize the loss of data.



Copyright © 2020, Oracle and/or its affiliates.

The database administrator (DBA) is typically responsible for ensuring that the database is open and available when users need it. To achieve that goal, the DBA (working with the system administrator):

- Anticipates and works to protect the database from failure wherever possible and to avoid common causes of failure
- Works to increase the mean time between failures (MTBF) that negatively affect availability
- Ensures that hardware is as reliable as possible, that critical components are protected by redundancy, and that operating system maintenance is performed in a timely manner. Oracle Database provides advanced configuration options to increase MTBF, including:
 - Real Application Clusters
 - Oracle Data Guard
- Decreases the mean time to recover (MTTR) by practicing recovery procedures in advance and configuring backups so that they are readily available when needed
- Minimizes the loss of data. DBAs who follow accepted best practices can configure their databases so that no committed transaction is ever lost. Entities that assist in guaranteeing this include:
 - Archive log files (discussed later in this lesson)
 - Flashback technology
 - Standby databases and Oracle Data Guard

Separation of DBA Duties

The SYSBACKUP administrative privilege:

- Includes permissions for backup and recovery (connecting to a closed database)
- Does not include data access privileges such as SELECT ANY TABLE
- Is granted to the SYSBACKUP user that is created during database installation
- Can be explicitly used in RMAN connections by a SYSBACKUP privileged user

```
$ rman target ''/ as sysbackup"  
connected to target database: ORCL (DBID=1297344416)
```



Copyright © 2020, Oracle and/or its affiliates.

Oracle Database provides support for separation of database administration (DBA) duties for the Oracle database with task-specific and least-privileged administrative privileges that do not require the SYSDBA administrative privilege. The privilege to connect and execute commands in Recovery Manager (RMAN) is the SYSBACKUP privilege.

- Explicitly connect with the SYSDBA role:

```
rman target ''/ as sysbackup''
```

Note the single quotation marks within the double quotation marks.

- For backward compatibility, rman target / connects as SYSDBA.

Assessing Your Recovery Requirements

- Identify and prioritize critical data.
- Base recovery requirements on data criticality.
 - Recovery Point Objective (RPO): Tolerance for data loss
 - How frequently should backups be taken?
 - Is point-in-time recovery required?
 - Recovery Time Objective (RTO): Tolerance for down time
 - Downtime: Problem identification + recovery planning + systems recovery
 - Tiered RTO per level of granularity (database, tablespace, table, row)
 - Determine backup retention policy for on-site, off-site, and long-term backups.
- Assess data protection requirements.
 - Physical: Disasters, outages, failures, corruptions
 - Logical: Human errors, application errors



Copyright © 2020, Oracle and/or its affiliates.

To assess your recovery requirements properly, you should first determine how critical a database that is lost or unavailable is to your business. Consider the following:

- How much will the company lose per hour of down time?
- How much will be lost in productivity and labor costs if the database is down?

By quantifying these costs, you will be able to justify hardware and storage expenditures to prevent and recover from database failures.

Your next step is to classify your databases according to their criticality. As an example, a company has a large data warehousing reporting system that can tolerate 12 hours worth of lost data, because batch loads can be rerun with a few hours of downtime that is acceptable to the user community. A disk and tape backup strategy may be appropriate for this type of system.

A company has a critical OLTP system that can tolerate no more than a few minutes worth of lost data and several minutes of acceptable down time, because a down database translates into \$100,000 per hour of lost revenue. For this system, a traditional backup and recovery solution will not suffice. The company needs to consider a “minimal downtime solution” such as Oracle Data Guard.

Determine your *Recovery Point Objective (RPO)*, which helps you evaluate how much data loss is acceptable. Consider the following:

- Are there some databases that require more frequent backups?
- Is adequate disk space available for archived logs (for point-in-time recovery as an example)?

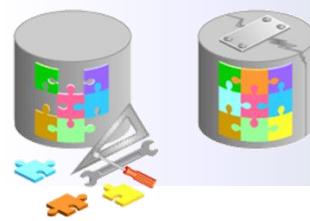
Consider your *Recovery Time Objective (RTO)* to determine how much recovery time you can tolerate. Is it hours or only minutes? Remember that RTO may vary by database and/or server. You may require a combined disk and tape backup strategy for critical databases to meet more pressing RTO requirements. When you have determined your desired recovery capability from disk and tape, assess the retention period for backups.

In addition to assessing your recovery requirements, you need to determine which failures you need to protect against. Consider physical losses of data and logical errors that can occur in your application and database.

Categories of Failure

Failures can generally be divided into the following categories:

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



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Failures can generally be divided into the following categories:

- **Statement failure:** A single database operation (select, insert, update, or delete) fails.
- **User process failure:** A single database session fails.
- **Network failure:** Connectivity to the database is lost.
- **User error:** A user successfully completes an operation, but the operation (dropping a table or entering bad data) is incorrect.
- **Instance failure:** The database instance shuts down unexpectedly.
- **Media failure:** A loss of any file that is needed for database operation (that is, the files have been deleted or the disk has failed).

Statement Failure

Typical Problems	Possible Solutions
Attempts to enter invalid data into a table	Work with users to validate and correct data.
Attempts to perform operations with insufficient privileges	Provide the appropriate object or system privileges.
Attempts to allocate space that fails	Enable resumable space allocation. Increase owner quota. Add space to the tablespace.
Logic errors in applications	Work with developers to correct program errors.



Copyright © 2020, Oracle and/or its affiliates.

When a single database operation fails, DBA involvement may be necessary to correct errors with user privileges or database space allocation. DBAs may also need to assist in troubleshooting, even for problems that are not directly in their task area. This can vary greatly from one organization to another.

For example, in organizations that use off-the-shelf applications (that is, organizations that have no software developers), the DBA is the only point of contact and must examine logic errors in applications.

To understand logic errors in applications, you should work with developers to understand the scope of the problem. Oracle Database tools may provide assistance by helping to examine audit trails or previous transactions.

Note: In many cases, statement failures are by design and desired. For example, security policies and quota rules are often decided upon in advance. If a user gets an error while trying to exceed his or her limits, it may be better for the operation to fail and no resolution may be necessary.

User Process Failure

Typical Problems	Possible Solutions
A user performs an abnormal disconnect.	A DBA's action is not usually needed to resolve user process failures.
A user's session is abnormally terminated.	Instance background processes roll back uncommitted changes and release locks.
A user experiences a program error that terminates the session.	The DBA should watch for trends.



Copyright © 2020, Oracle and/or its affiliates.

User processes that abnormally disconnect from the instance may have uncommitted work in progress that needs to be rolled back. The Process Monitor (PMON) background process periodically polls server processes to ensure that their sessions are still connected. If PMON finds a server process whose user is no longer connected, PMON recovers from any ongoing transactions; it also rolls back uncommitted changes and releases any locks that are held by the failed session.

A DBA's intervention should not be required to recover from user process failure, but the administrator must watch for trends. One or two users disconnecting abnormally is not a cause for concern. A small percentage of user process failures may occur from time to time.

But consistent and systemic failures indicate other problems. A large percentage of abnormal disconnects may indicate a need for user training (which includes teaching users to log out rather than just terminate their programs). It may also be indicative of network or application problems.

Network Failure

Typical Problems	Possible Solutions
Listener fails	Configure a backup listener and connect-time failover.
Network interface card (NIC) fails	Configure multiple network cards.
Network connection fails	Configure a backup network connection.



Copyright © 2020, Oracle and/or its affiliates.

The best solution to network failure is to provide redundant paths for network connections. Backup listeners, network connections, and network interface cards reduce the chance that network failures will affect system availability.

User Error

Typical Problems	Possible Solutions
User inadvertently deletes or modifies data	Roll back a transaction and dependent transactions or rewind the table
User drops a table	Recover the table from recycle bin Recover the table from a backup

Use Oracle LogMiner to query your online redo logs and archived redo logs through an Enterprise Manager or SQL interface.



Copyright © 2020, Oracle and/or its affiliates.

Users may inadvertently delete or modify data. If they have not yet committed or exited their program, they can simply roll back.

You can use Oracle LogMiner to query your online redo logs and archived redo logs through an Enterprise Manager or SQL interface. Transaction data may persist in online redo logs longer than it persists in undo segments; if you have configured archiving of redo information, redo persists until you delete the archived files. Oracle LogMiner is discussed in *Oracle Database Utilities*.

Users who drop a table can recover it from the recycle bin by flashing back the table to before the drop.

If the recycle bin has already been purged, or if the user dropped the table with the PURGE option, the dropped table can still be recovered by using point-in-time recovery (PITR) if the database has been properly configured.

RMAN enables you to recover one or more tables or table partitions to a specified point in time without affecting the remaining database objects.

Note: Flashback technologies, PITR, and table recovery are discussed in the Oracle Database 18c: Backup and Recovery Workshop course and in *Oracle Database Backup and Recovery User's Guide*.

Instance Failure

Typical Causes	Possible Solutions
Power outage	Restart the instance by using the STARTUP command. Recovering from instance failure is automatic, including rolling forward changes in the redo logs and then rolling back any uncommitted transactions.
Hardware failure	Investigate the causes of failure by using the alert log, trace files, and Enterprise Manager.
Failure of one of the critical background processes	
Emergency shutdown procedures	



Copyright © 2020, Oracle and/or its affiliates.

Instance failure occurs when the database instance is shut down before synchronizing all database files. An instance failure can occur because of hardware or software failure or through the use of the emergency SHUTDOWN ABORT and STARTUP FORCE shutdown commands.

Administrator involvement in recovering from instance failure is rarely required if Oracle Restart is enabled and is monitoring your database. Oracle Restart attempts to restart your database instance as soon as it fails. If manual intervention is required, then there may be a more serious problem that prevents the instance from restarting, such as a memory CPU failure.

Media Failure

Typical Causes	Possible Solution
Failure of a disk drive	1. Restore the affected file from backup.
Failure of a disk controller	2. Inform the database server of a new file location (if necessary).
Deletion or corruption of a file needed for a database operation	3. Recover the file by applying redo information (if necessary).
Storage network failure	
Solid state storage corruption	



Copyright © 2020, Oracle and/or its affiliates.

Oracle Corporation defines media failure as any failure that results in the loss or corruption of one or more database files (data, control, or redo log file).

Recovering from media failure requires that you restore and recover the missing files.

Data Failures

- **Inaccessible components:** Missing data files at the OS level, incorrect access permissions, offline tablespace
- **Physical corruptions:** Block checksum failures, invalid block header field values
- **Logical corruptions:** Inconsistent dictionary, corrupt row piece, index entry, or transaction
- **Inconsistencies:** Control file older or newer than the data files and online redo logs
- **I/O failures:** Limit on the number of open files exceeded, inaccessible channels, network or I/O error



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The slide lists the additional types of failures you should anticipate in your environment.

Instance Recovery

You can understand instance recovery by becoming familiar with the following concepts and procedures:

- The checkpoint (CKPT) process
- Redo log files and the Log Writer (LGWR) process
- Automatic instance or crash recovery
- Phases of instance recovery
- Tuning instance recovery
- Using the MTTR Advisor



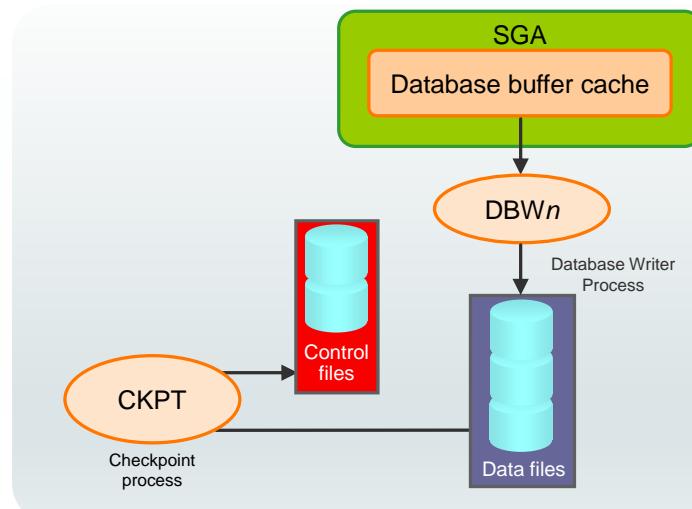
Copyright © 2020, Oracle and/or its affiliates.

These topics are discussed in detail in the following slides.

The Checkpoint (CKPT) Process

CKPT is responsible for:

- Updating data file headers with checkpoint information
- Updating control files with checkpoint information
- Signaling DBW n at full checkpoints



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

To understand instance recovery, you need to understand the functioning of certain background processes.

Every three seconds (or more frequently), the CKPT process stores data in a control file to document the modified data blocks that DBW n has written from the SGA to disk. This is called an “incremental checkpoint.” The purpose of a checkpoint is to identify that place in the online redo log file where instance recovery is to begin (which is called the “checkpoint position”).

In the event of a log switch, the CKPT process also writes this checkpoint information to the headers of data files.

Checkpoints exist for the following reasons:

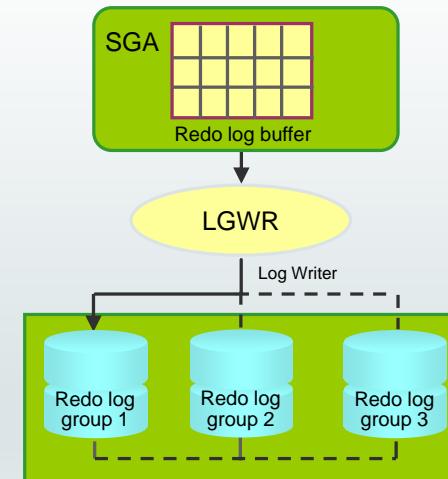
- To ensure that modified data blocks in memory are written to the disk regularly so that data is not lost in case of a system or database failure
- To reduce the time required for instance recovery (only the online redo log file entries following the last checkpoint need to be processed for recovery)
- To ensure that all committed data has been written to data files during shutdown

The checkpoint information written by the CKPT process includes checkpoint position, system change number (SCN), location in the online redo log file to begin recovery, information about logs, and so on.

Note: The CKPT process does not write data blocks to the disk or redo blocks to the online redo log files.

Redo Log Files and the Log Writer (LGWR) Process

- Redo log files:
 - Record changes to the database
 - Should be multiplexed to protect against loss
- Log Writer (LGWR) writes:
 - At commit
 - When the redo log buffer is one-third full
 - Every three seconds
 - Before DBWn writes
 - Before clean shutdowns



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

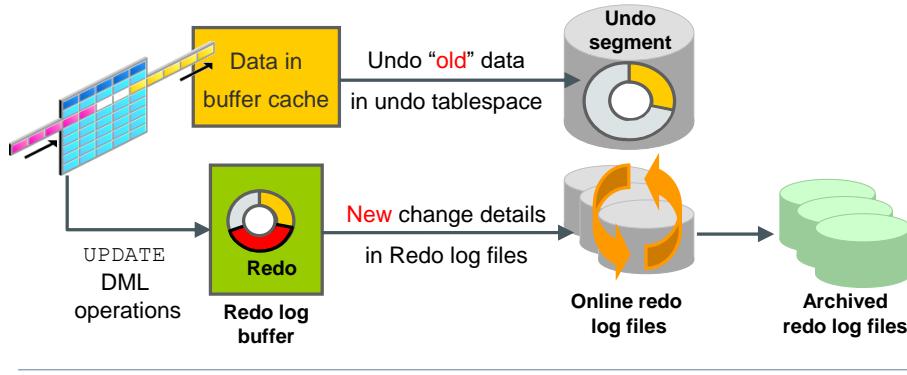
Redo log files record changes to the database as a result of transactions and internal Oracle server actions. A transaction is a logical unit of work consisting of one or more SQL statements. Redo log files protect the database from loss of integrity because of system failures caused by power outages, disk failures, and so on. Redo log files should be multiplexed to ensure that the information stored in them is not lost in the event of a disk failure.

The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group, and each group is identified by a number. The Log Writer (LGWR) process writes redo records from the redo log buffer to all members of a redo log group until the files are filled or a log switch operation is requested.

It then switches and writes to the files in the next group. Redo log groups are used in a circular fashion.

Best practice tip: If possible, multiplexed redo log files should reside on different disks.

Database Log Mode



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

As modifications are made to data in the database, the “old” data is stored in the undo tablespace and the new change details in online redo log files (as shown in the graphic). The undo is also in the redo stream. Contents of the online redo log include uncommitted transactions and schema and object management statements. The database maintains online redo log files to protect against data loss. Specifically, after an instance failure the online redo log files enable Oracle Database to recover committed data that it has not yet written to the data files.

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

It is possible to perform any type of backup (full or incremental) of a database that is in **NOARCHIVELOG mode**—if, of course, the database is not open. When the database is in **NOARCHIVELOG mode**, you must restore all data files before executing a recovery operation. Note also that recovery is limited to the time of the last backup. The database can be recovered to the last committed transaction only when the database is in **ARCHIVELOG mode**.

Use cases for both log modes are included in the table in the slide.

Automatic Instance Recovery or Crash Recovery

- Is caused by attempts to open a database whose files are not synchronized on shutdown
- Uses information stored in redo log groups to synchronize files
- Involves two distinct operations:
 - **Rolling forward:** Redo log changes (both committed and uncommitted) are applied to data files.
 - **Rolling back:** Changes that are made but not committed are returned to their original state.



Copyright © 2020, Oracle and/or its affiliates.

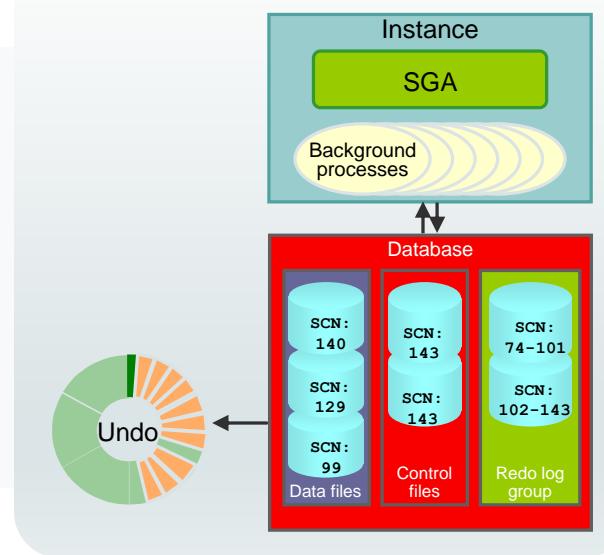
The Oracle Database server automatically recovers from instance failure. All that needs to happen is for the instance to be started normally. The instance mounts the control files and then attempts to open the data files. When it discovers that the data files have not been synchronized during shutdown, the instance uses information contained in the redo log groups to roll the data files forward to the time of shutdown. Then the database is opened, and any uncommitted transactions are rolled back.

Crash recovery and instance recovery are defined as follows:

- **Crash recovery:** The automatic application of online redo records to a database after either a single instance database crashes or all instances of an Oracle Real Applications Cluster configuration crash. Crash recovery only requires redo from the online logs; archived redo logs are not required. Often this is referred to as “instance recovery.”
- **Instance recovery:** In an Oracle RAC configuration, the application of redo data to an open database by an instance when this instance discovers that another instance has crashed

Phases of Instance Recovery

1. Instance startup (data files are out of sync)
2. Roll forward (redo)
3. Committed and uncommitted data in files
4. Database opened
5. Roll back (undo)
6. Committed data in files



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

For an instance to open a data file, the system change number (SCN) contained in the data file's header must match the current SCN that is stored in the database's control files.

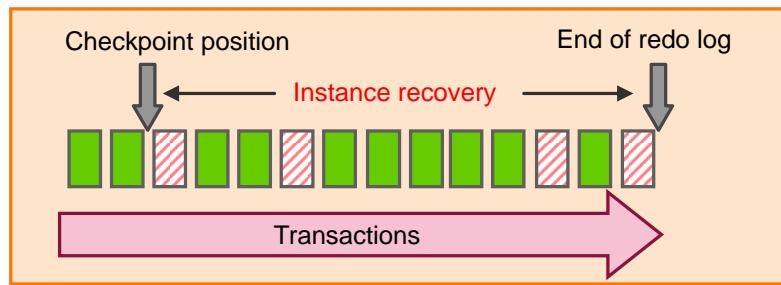
If the numbers do not match, the instance applies redo data from the online redo logs, sequentially "redoing" transactions until the data files are up-to-date. After all data files have been synchronized with the control files, the database is opened, and users can log in.

When redo logs are applied, *all* transactions are applied to bring the database up to the state as of the time of failure. This usually includes transactions that are in progress but have not yet been committed. After the database has been opened, those uncommitted transactions are rolled back. Blocks from uncommitted transactions remain locked until the undo has been applied and committed.

At the end of the rollback phase of instance recovery, the data files contain only committed data.

Tuning Instance Recovery

- During instance recovery, the transactions between the checkpoint position and the end of the redo log must be applied to data files.
- You tune instance recovery by controlling the difference between the checkpoint position and the end of the redo log.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Transaction information is recorded in the redo log groups before the instance returns “commit complete” for a transaction. The information in the redo log groups guarantees that the transaction can be recovered in case of a failure. The transaction information must also be written to the data file. The data file write usually happens at some time after the information is recorded in redo log groups because the data file write process is much slower than the redo writes. Random writes for data files are slower than serial writes for redo log files.

Every three seconds, the checkpoint process records information in the control file about the checkpoint position in the redo log. Therefore, the Oracle Database server knows that all redo log entries recorded before this point are not necessary for database recovery. In the graphic in the slide, the striped blocks have not yet been written to the disk.

The time required for instance recovery is the time required to bring data files from their last checkpoint to the latest SCN recorded in the control file. The administrator controls that time by setting an MTTR target (in seconds) and through the sizing of redo log groups. For example, for two redo groups, the distance between the checkpoint position and the end of the redo log group cannot be more than 90% of the smallest redo log group.

Using the MTTR Advisor

- Specify the desired time in seconds or minutes.
- The default value is 0 (disabled).
- The maximum value is 3,600 seconds (one hour).

The screenshot shows the 'Recovery Settings' page in Oracle Enterprise Manager Cloud Control. At the top right, it says 'Logged in as DBA1'. Below that are three buttons: 'Show SQL', 'Revert', and 'Apply'. The main section is titled 'Instance Recovery'. It contains a note about the fast-start checkpointing feature and its relation to the `FAST_START_MTTR_TARGET` parameter. A text input field shows 'Current Estimated Mean Time To Recover (seconds) 19'. Below it is another input field labeled 'Desired Mean Time To Recover' with a value of '0' and a dropdown menu set to 'Minutes'.

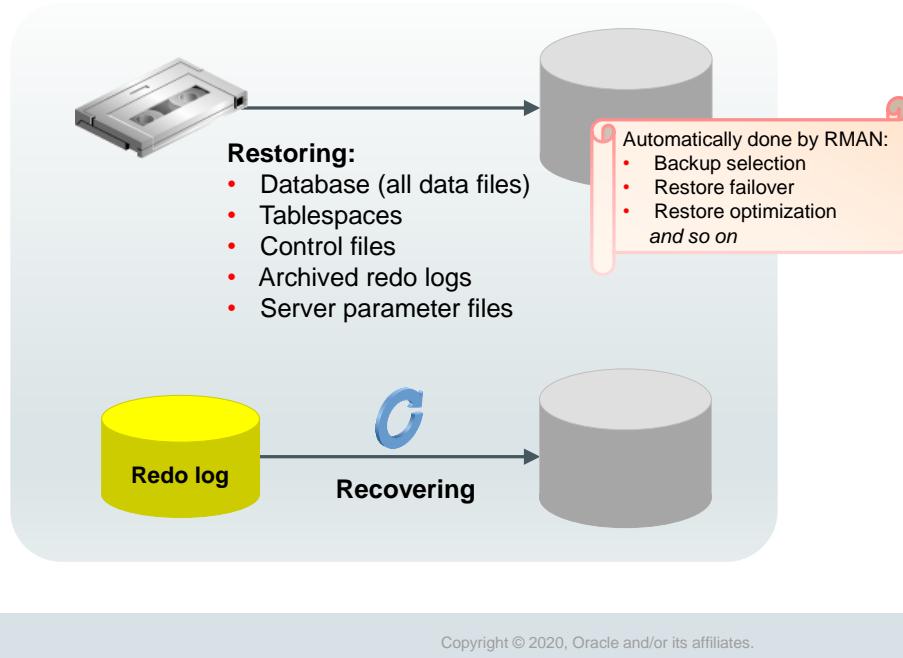
The `FAST_START_MTTR_TARGET` initialization parameter simplifies the configuration of recovery time from instance or system failure. The MTTR Advisor converts the `FAST_START_MTTR_TARGET` value into several parameters to enable instance recovery in the desired time (or as close to it as possible). Note that explicitly setting the `FAST_START_MTTR_TARGET` parameter to 0 disables the MTTR Advisor.

The `FAST_START_MTTR_TARGET` parameter must be set to a value that supports the service level agreement for your system. A small value for the MTTR target increases I/O overhead because of additional data file writes (affecting the performance). However, if you set the MTTR target too large, the instance takes longer to recover after a crash.

For assistance in setting the MTTR target by using Enterprise Manager Cloud Control, navigate as follows:

- Performance > Advisors Home > MTTR Advisor
- Availability > Backup & Recovery > Recovery Settings

Restoring and Recovering



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The “recovery unit” includes two major types of activities: restoring and recovering.

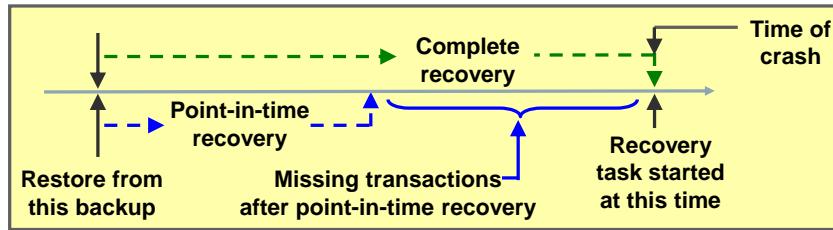
- *Restoring* a file is the process of copying a backup into place to be used by the database. This is necessary if, for example, a file is damaged because the physical disk it is on fails. This is usually due to hardware problems, such as disk write errors or controller failure. In that case, a backup of the file needs to be copied on to a new (or repaired) disk. The file types that can be restored are listed in the slide.
 - RMAN uses the records of available backup sets or image copies in the RMAN repository to select the best available backups. If two backups are from the same point in time, then RMAN prefers image copies over backup sets because RMAN can restore them more quickly (similar for disk versus tape).
 - RMAN automatically uses restore failover to skip corrupted or inaccessible backups and look for usable backups.
 - By default, RMAN skips restoring a data file if the file is present in the correct location and its header contains the expected information, and so on.
- *Recovering* the file entails applying redo such that the state of the file is brought forward in time, to whatever point you want. That point is usually as close to the time of failure as possible.

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

Comparing Complete and Incomplete Recovery

Recovery can have two kinds of scope:

- **Complete recovery:** Brings the database or tablespace up to the present, including all committed data changes made to the point in time when the recovery was requested
- **Incomplete or point-in-time recovery (PITR):** Brings the database or tablespace up to a specified point in time in the past, before the recovery operation was requested



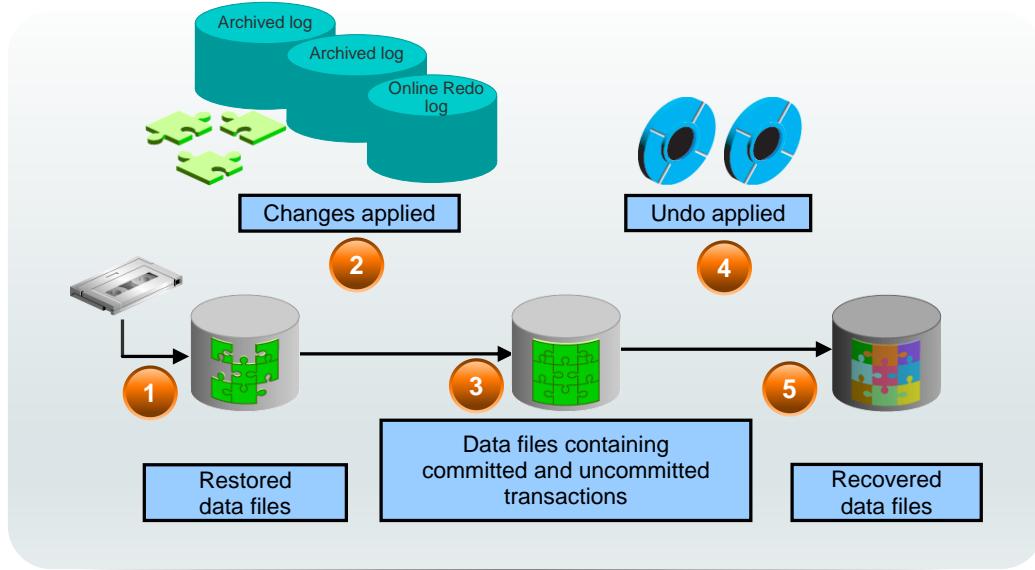
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

When you perform complete recovery, you bring the database to the state where it is fully up-to-date, including all committed data modifications to the present time.

Incomplete recovery, however, brings the database or tablespace to some point of time in the past. This is also known as “point-in-time recovery (PITR).” It means there are missing transactions; any data modifications done between the recovery destination time and the present are lost. In many cases, this is the desirable goal because there may have been some changes made to the database that need to be undone. Recovering to a point in the past is a way to remove the unwanted changes.

The Complete Recovery Process



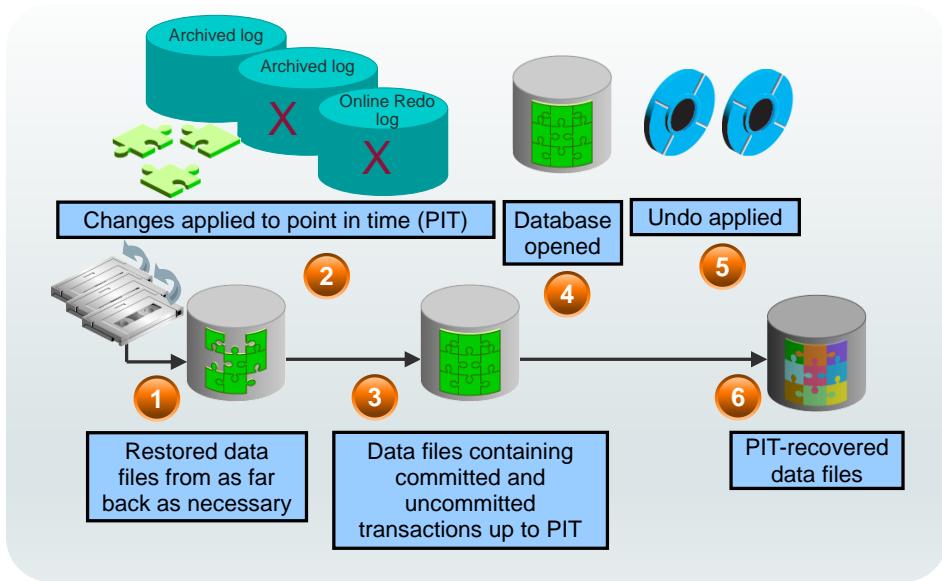
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The following steps describe what takes place during complete recovery:

1. Damaged or missing files are restored from a backup.
2. Changes from incremental backups, archived redo log files, and online redo log files are applied as necessary. The redo log changes are applied to the data files until the current online log is reached and the most recent changes have been reapplied. Undo blocks are generated during this entire process. This is referred to as rolling forward or cache recovery.
3. The restored data files may now contain committed and uncommitted changes.
4. The undo blocks are used to roll back any uncommitted changes. This is sometimes referred to as transaction recovery.
5. The data files are now in a recovered state and are consistent with the other data files in the database.

The Point-in-Time Recovery Process



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Incomplete recovery, or database point-in-time recovery (DBPITR), uses a backup to produce a noncurrent version of the database. That is, you do not apply all the redo records generated after the most recent backup. Perform this type of recovery only when absolutely necessary. To perform point-in-time recovery, you need:

- A valid offline or online backup of all the data files made before the recovery point
- All archived logs from the time of the backup until the specified time of recovery

The steps to perform a point-in-time recovery are as follows:

1. **Restore the data files from backup:** The backup that is used must be from before your target recovery point. This entails either copying files using OS commands or using the RMAN RESTORE command.
2. **Use the RECOVER command:** Apply redo from the archived redo log files, including as many as necessary to reach the restore point destination.
3. **State of over-recovery:** Now the data files contain some committed and some uncommitted transactions because the redo can contain uncommitted data.
4. **Use the ALTER DATABASE OPEN command:** The database is opened before undo is applied. This is to provide higher availability.
5. **Apply undo data:** While the redo was being applied, redo supporting the undo data files was also applied. So the undo is available to be applied to the data files in order to undo any uncommitted transactions. That is done next.
6. **Process complete:** The data files are now recovered to the point in time that you chose.

Oracle Flashback Database is the most efficient alternative to DBPITR. Unlike the other flashback features, it operates at a physical level and reverts the current data files to their contents at a past time. The result is like the result of a DBPITR, including the OPEN RESETLOGS, but Flashback Database is typically faster because it does not require you to restore data files and requires only limited application of redo compared to media recovery.

For Instructor Use Only.
This document should not be distributed.

Oracle Data Protection Solutions

Backup and Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Hours/Days	Recovery Manager Oracle Secure Backup
Logical data protection	Minutes/Hours	Flashback Technologies
Recovery analysis	Minimize time for problem identification and recovery planning	Data Recovery Advisor

Disaster Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Seconds/Minutes	Data Guard Active Data Guard



Copyright © 2020, Oracle and/or its affiliates.

Oracle provides an appropriate data protection solution depending on your backup and recovery objective and RTO:

- Oracle Recovery Manager (RMAN) is the core Oracle Database software component that manages database backup, restore, and recovery processes.
- Oracle Secure Backup (OSB) is Oracle's enterprise-grade tape backup management solution for both database and file system data.
- Oracle Database Flashback technologies are a set of data recovery solutions that enable human errors to be reversed by selectively and efficiently undoing the effects of a mistake.
- The Data Recovery Advisor provides intelligent database problem identification and recovery capabilities.
- Data Guard and Active Data Guard enable physical standby databases to be open for read access while being kept synchronized with the production database through media recovery.

Flashback Technology

Use Flashback technology for:

- Viewing past states of data
- Winding data back and forth in time
- Assisting users in error analysis and recovery



For error analysis:

Oracle Flashback Query

Oracle Flashback Versions Query

Oracle Flashback Transaction Query

For error recovery:

Oracle Flashback Transaction Backout

Oracle Flashback Table

Oracle Flashback Drop

Oracle Flashback Database



Copyright © 2020, Oracle and/or its affiliates.

Oracle Database includes Oracle Flashback technology: a group of features that support viewing past states of data and winding data back and forth in time, without requiring the restoration of the database from backup. With this technology, you help users analyze and recover from errors. For users who have committed erroneous changes, use the following to analyze the errors:

- **Flashback Query:** View committed data as it existed at some point in the past. The SELECT command with the AS OF clause references a time in the past through a time stamp or system change number (SCN).
- **Flashback Version Query:** View committed historical data for a specific time interval. Use the VERSIONS BETWEEN clause of the SELECT command (for performance reasons with existing indexes).
- **Flashback Transaction Query:** View all database changes made at the transaction level.

Possible solutions to recover from user error:

- **Flashback Transaction Backout:** Rolls back a specific transaction and dependent transactions
- **Flashback Table:** Rewinds one or more tables to their contents at a previous time without affecting other database objects

Summary

In this lesson, you should have learned how to:

- Identify the responsibilities of a DBA in database backup and recovery
- Identify the types of failure that can occur in an Oracle database
- Describe instance recovery
- Describe complete and incomplete recovery



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Backup and Recovery Configuration

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- Configure the fast recovery area
- Multiplex the control file
- Multiplex redo log files
- Configure ARCHIVELOG mode

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Configuring for Recoverability

Configure your database for maximum recoverability by:

- Scheduling regular backups
- Multiplexing control files
- Multiplexing redo log groups
- Retaining archived copies of redo logs



Copyright © 2020, Oracle and/or its affiliates.

To provide the best protection for your data, you must:

- **Schedule regular backups:** Most media failures require that you restore the lost or damaged file from backup.
- **Multiplex control files:** All control files associated with a database are identical. Recovering from the loss of a single control file is not difficult; recovering from the loss of *all* control files is much more challenging. Guard against losing all control files by having at least two copies.
- **Multiplex redo log groups:** To recover from instance or media failure, redo log information is used to roll data files forward to the last committed transaction. If your redo log groups rely on a single redo log file, the loss of that file means that data is likely to be lost. Ensure that there are at least two copies of each redo log group; if possible, each copy should be under different disk controllers.
- **Retain archived copies of redo logs:** If a file is lost and restored from backup, the instance must apply redo information to bring that file up to the latest SCN contained in the control file. With the default setting, the database can overwrite redo information after it has been written to the data files. Your database can be configured to retain redo information in archived copies of the redo logs. This is known as placing the database in ARCHIVELOG mode.

You can perform configuration tasks in Enterprise Manager Cloud Control or by using the SQL command line.

Configuring the Fast Recovery Area

- Fast recovery area:
 - Strongly recommended for simplified backup storage management
 - Storage space (separate from working database files)
 - Location specified by the `DB_RECOVERY_FILE_DEST` parameter
 - Size specified by the `DB_RECOVERY_FILE_DEST_SIZE` parameter
 - Large enough for backups, archived logs, flashback logs, multiplexed control files, and multiplexed redo logs
 - Automatically managed according to your retention policy
- Configuration of the fast recovery area includes specifying the location, size, and retention policy.

```
ALTER SYSTEM SET db_recovery_file_dest = directory | disk group
ALTER SYSTEM SET db_recovery_file_destsize = integer [K | M | G]
```



Copyright © 2020, Oracle and/or its affiliates.

The fast recovery area is space that is set aside on disk to contain archived logs, backups, flashback logs, multiplexed control files, and multiplexed redo logs. A fast recovery area simplifies backup storage management and is strongly recommended. You should place the fast recovery area on storage space that is separate from the location of your database data files, primary online log files, and control file.

The amount of disk space to allocate for the fast recovery area depends on the size and activity levels of your database. As a general rule, the larger the fast recovery area, the more useful it is. Ideally, the fast recovery area should be large enough for copies of your data and control files and for flashback, online redo, and archived logs needed to recover the database with the backups kept based on the retention policy. In short, the fast recovery area should be at least twice the size of the database so that it can hold one backup and several archived logs.

Space management in the fast recovery area is governed by a backup retention policy. A retention policy determines when files are obsolete, which means that they are no longer needed to meet your data recovery objectives. The Oracle Database server automatically manages this storage by deleting files that are no longer needed.

Monitoring the Fast Recovery Area

- Monitor the fast recovery area to ensure that it does not reach its capacity.
- The instance will pause if there isn't enough space in the fast recovery area to create an archived log.
- Query the `V$RECOVERY_FILE_DEST` view to determine the current location, disk quota, space in use, space reclaimable by deleting files, and total number of files in the fast recovery area.
- Query the `V$RECOVERY_AREA_USAGE` view to determine the percentage of the total disk quota used by different types of files.
- You can also use GUI tools such as Enterprise Manager Cloud Control to monitor the space usage.



Copyright © 2020, Oracle and/or its affiliates.

If you have configured your archived logs to be written to this location, it is important to monitor this space to ensure that it does not reach its capacity. If the instance is unable to create an archived log because of lack of space, it pauses until the administrator corrects the situation.

You can use the `V$RECOVERY_FILE_DEST` and `V$RECOVERY_AREA_USAGE` views to determine whether you have allocated enough space for your fast recovery area.

The retention time determines when files are obsolete (that is, when they are no longer needed to meet your data recovery objectives). The Oracle Database server automatically manages this storage, deleting files that are no longer needed. You can back up the recovery area so that Oracle Recovery Manager (RMAN) can fail over to other archived redo log destinations if the archived redo log in the fast recovery area is inaccessible or corrupted.

Periodically copying backups to tape frees space in the fast recovery area for other files, but retrieving files from tape causes longer database restoration and recovery times.

Multiplexing Control Files

To protect against database failure, your database should have multiple copies of the control file.

	ASM Storage	File System Storage
Best Practice	One copy on each disk group (such as +DATA and +FRA)	At least two copies, each on a separate disk (at least one on a separate disk controller)
Steps to create additional control files	No additional control file copies required	<ol style="list-style-type: none">1. Alter the SPFILE with the ALTER SYSTEM SET control_files command.2. Shut down the database.3. Copy the control file to a new location.4. Open the database and verify the addition of the new control file.



Copyright © 2020, Oracle and/or its affiliates.

A control file is a binary file that describes the structure of the database. It must be available for writing by the Oracle server whenever the database is mounted or opened. Without this file, the database cannot be mounted, and recovery or re-creation of the control file is required. Your database should have a minimum of two control files on different storage devices to minimize the impact of a loss of one control file.

The loss of a single control file causes the instance to fail because all control files must be available at all times. However, recovery can be a simple matter of copying one of the other control files. The loss of all control files is slightly more difficult to recover from but is not usually catastrophic.

Adding a Control File

In a database using regular file system storage, adding a control file is a manual operation:

1. Alter the SPFILE with the following command specifying the appropriate location of your files:

```
ALTER SYSTEM SET control_files =
  '/u01/app/oracle/oradata/orcl/control01.ctl' ,
  '/u02/app/oracle/oradata/orcl/control02.ctl' ,
  '/u03/app/oracle/oradata/orcl/control03.ctl' SCOPE=SPFILE;
```

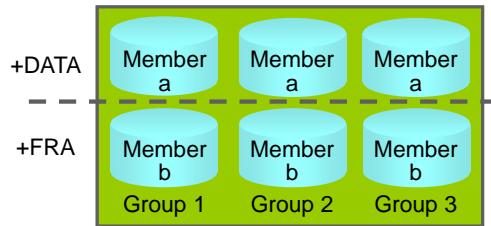
2. Shut down the database instance.
3. Use an operating system command to copy an existing control file to the location you select for your new file.
4. Open the database.

If you are using ASM as your storage technique, then as long as you have two control files, one in each disk group (such as +DATA and +FRA), you should not require further multiplexing. In a database using Oracle Managed Files (OMF)—such as a database using ASM storage—all additional control files must be created as part of a recovery process using RMAN (or using Enterprise Manager).

Redo Log Files

Multiplex redo log groups to protect against media failure and loss of data. This increases database I/O. It is suggested that redo log groups have:

- At least two members (files) per group
- Each member:
 - On a separate disk or controller if using file system storage
 - In a separate disk group (such as +DATA and +FRA) if using ASM



Note: Multiplexing redo logs may impact overall database performance.



Copyright © 2020, Oracle and/or its affiliates.

Redo log groups are made up of one or more redo log files. Each log file in a group is a duplicate of the others. Oracle Corporation recommends that redo log groups have at least two files per group. If you are using file system storage, then each member should be distributed on separate disks or controllers so that no single equipment failure impacts an entire log group. If you are using ASM storage, then each member should be in a separate disk group, such as +DATA and +FRA.

The loss of an entire current log group is one of the most serious media failures because it can result in loss of data. The loss of a single member of a multiple-member log group is trivial and does not affect database operation (other than causing an alert to be published in the alert log). Recovery from the loss of an entire log group requires advanced recovery techniques and is discussed in the course titled *Oracle Database 18c: Backup and Recovery Workshop*.

Remember that multiplexing redo logs may heavily influence database performance because a commit cannot complete until the transaction information has been written to the logs. You must place your redo log files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files (unless you are using ASM). Because only one group is written to at a given time, there is no performance impact in having members from several groups on the same disk.

For information about file placement in a Database Cloud Service database deployment, see “Characteristics of a Newly Created Deployment” in *Administering Oracle Database Cloud Service*.

Multiplexing the Redo Log

Add a member to an existing log group:

- Navigate to the Redo Log Groups page in Enterprise Manager Database Express.
- Use the `ALTER DATABASE` command:

```
SQL> ALTER DATABASE
  2 ADD LOGFILE MEMBER '/u01/app/oracle/oradata/orcl/redo1a.log'
  3 TO GROUP 1;
```



Copyright © 2020, Oracle and/or its affiliates.

You can multiplex your redo log by adding a member to an existing log group. To add a member to a redo log group (with open database and no impact on user performance), perform the following steps in Enterprise Manager Database Express:

1. Select Storage > Redo Log Groups.
2. Select a group and click Add Member.
The Add Member page appears.
3. For File System storage, enter the file name and the file directory. Click OK.

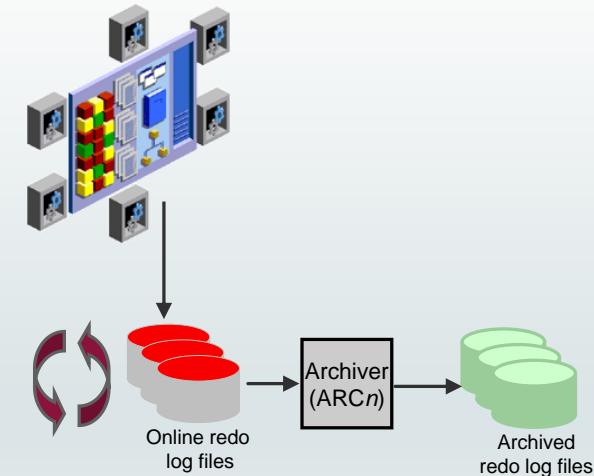
Repeat these steps for every existing group that you want to multiplex.

When you add the redo log member to a group, the member's status is marked as `INVALID` (as can be seen in the `V$LOGFILE` view). This is the expected state because the new member of the group has not yet been written to. When a log switch occurs and the group containing the new member becomes `CURRENT`, the member's status changes to null.

Creating Archived Redo Log Files

To preserve redo information, create archived copies of redo log files by performing the following steps:

1. Specify the archived redo log file-naming convention.
2. Specify one or more archived redo log file locations.
3. Place the database in ARCHIVELOG mode.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The Oracle Database server treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next. After all groups have been written to, the Oracle Database server begins overwriting information in the first log group.

To configure your database for maximum recoverability, you must instruct the Oracle Database server to make a copy of the online redo log group before allowing it to be overwritten. These copies are known as *archived redo log files*.

To facilitate the creation of archived redo log files:

1. Specify a naming convention for your archived redo log files.
2. Specify a destination or destinations for storing your archived redo log files.
3. Place the database in ARCHIVELOG mode.

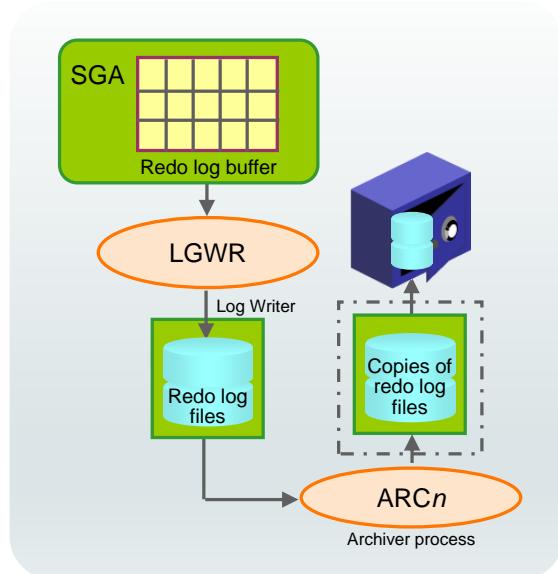
Note: Steps 1 and 2 are not necessary if you are using a fast recovery area.

The destination should exist before placing the database in ARCHIVELOG mode. When a directory is specified as a destination, there should be a slash at the end of the directory name.

Archiver (ARCn) Process

Archiver (ARCn):

- Automatically archives online redo log files when the database is in ARCHIVELOG mode
- Preserves a record of all changes made to the database



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

ARCn is a background process that is crucial to the recovery of a database after the loss of a disk. When an online redo log group gets filled, the Oracle Database server begins writing to the next online redo log group. The process of switching from one online redo log group to another is called a *log switch*. The ARCn process initiates archiving of the filled log group at every log switch. It automatically archives the online redo log group before the log group can be reused so that all the changes made to the database are preserved. This enables recovery of the database to the point of failure even if a disk drive is damaged.

One of the important decisions that a DBA must make is whether to configure the database to operate in ARCHIVELOG mode or in NOARCHIVELOG mode.

- In NOARCHIVELOG mode, the online redo log files are overwritten each time a log switch occurs.
- In ARCHIVELOG mode, inactive groups of filled online redo log files must be archived before they can be used again.

Notes

- ARCHIVELOG mode is essential for most backup strategies.
- If the archived redo log file destination fills up or cannot be written to, the database will eventually come to a halt. Remove archived redo log files from the archived redo log file destination and the database will resume operations.

Archived Redo Log Files: Naming and Destinations

- Use the `LOG_ARCHIVE_DEST` initialization parameter to specify a single destination.
- Use the `LOG_ARCHIVE_DEST_n` initialization parameters to archive to two or more locations.
- If you are using file system storage, it is recommended that you add multiple locations across different disks.
- If the fast recovery area is enabled, `USE_DB_RECOVERY_FILE_DEST` is specified by default as an archived redo log file destination.



Copyright © 2020, Oracle and/or its affiliates.

Each archived redo log file must have a unique name to avoid overwriting older log files. Specify the naming format as shown in the slide. To help create unique file names, Oracle Database allows several wildcard characters in the name format:

- `%s`: Includes the log sequence number as part of the file name
- `%t`: Includes the thread number as part of the file name
- `%r`: Includes the resetlogs ID to ensure that the archive log file name remains unique (even after certain advanced recovery techniques that reset log sequence numbers)
- `%d`: Includes the database ID as part of the file name

The format should include `%s`, `%t`, and `%r` as best practice. (`%d` can also be included if multiple databases share the same archive log destination.)

By default, if the fast recovery area is enabled, `USE_DB_RECOVERY_FILE_DEST` is specified as an archived redo log file destination. Archived redo log files can be written to as many as 31 different destinations. Destinations may be local (a directory) or remote (an Oracle Net alias for a standby database).

Configuring ARCHIVELOG Mode

- You can use SQL commands as follows:
 1. Shut down the database instance if it is open.
 2. Mount the database.
 3. Issue the ALTER DATABASE ARCHIVELOG command.
 4. Open the database.

```
SQL> shutdown immediate
SQL> startup mount
SQL> alter database archivelog;
SQL> alter database open;
```

- You can also use Enterprise Manager Cloud Control to place the database in ARCHIVELOG mode.



Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

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

Summary

In this lesson, you should have learned how to:

- Configure the Fast Recovery Area
- Multiplex the control file
- Multiplex redo log files
- Configure ARCHIVELOG mode



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Verifying that the Control File is Multiplexed
- Configuring the Size of the Fast Recovery Area
- Verifying that the Redo Log File Is Multiplexed
- Configuring ARCHIVELOG Mode



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

3

Using Recovery Manager (RMAN)



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- Describe the features and functions of Recovery Manager (RMAN)
- Configure and manage RMAN settings

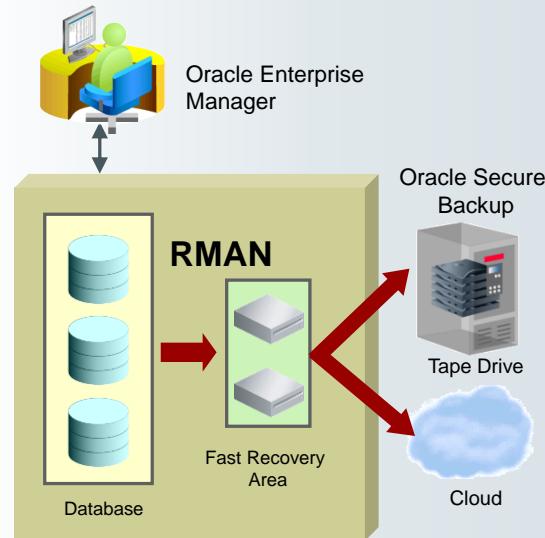
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Integrated Oracle Recovery Manager (RMAN)

- Intrinsic knowledge of database file formats and recovery procedures
 - Tablespace and data file recovery
 - Block validation
 - Online block-level recovery
 - Online, multistreamed backup
 - Unused block compression
 - Native encryption
- Integrated disk, tape, and cloud backup leveraging the fast recovery area and Oracle Secure Backup



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Oracle Recovery Manager (RMAN) is the core Oracle Database software component that manages database backup, restore, and recovery processes. RMAN maintains configurable backup and recovery policies and keeps historical records of all database backup and recovery activities. RMAN ensures that all files required to successfully restore and recover a database are included in complete database backups. Furthermore, as part of RMAN backup operations, all data blocks are verified to ensure that corrupt blocks are not propagated into the backup files.

The graphic in the slide provides one suggested configuration for backups. In this graphic, RMAN uses the fast recovery area for backups. Optionally, through Oracle Secure Backup, backups can be made to the tape drive and to the cloud. Backups can be performed to other designated locations, such as directly to disk.

Connecting to RMAN and a Target Database

```
. oraenv
orcl
$ rman target ''/ as sysbackup''

RMAN> BACKUP DATABASE;
Starting backup at 10-OCT-18
.
.

RMAN> LIST BACKUP;
BS Key Type LV Size    Device Type Elapsed Time Completion Time
----- -- -- -- -- -- -- -- -- 
1      Full   1.06G   DISK        00:01:49   10-OCT-18
.
.

RMAN> DELETE OBSOLETE;
.
Do you really want to delete the above objects (enter YES or NO)? YES
deleted archived log
.
.
```



Copyright © 2020, Oracle and/or its affiliates.

In your Linux course environment, you have more than one local database. Use `. oraenv` to set your environment variables (as shown in the slide).

Invoke RMAN at the operating system command line and specify the appropriate options. Commonly used options are:

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

The RMAN invocation `rman target /` connects to the local database as the target. The example in the slide shows the default login with the `SYSBACKUP` privilege.

At the RMAN prompt, you can submit RMAN commands to manage your backup environment and create backups in many different ways, depending on your needs. Shown in the slide are commands to perform a database backup, to list your existing backups (`LIST BACKUP`), and to delete any obsolete backups (`DELETE OBSOLETE`).

Note: Refer to *Oracle Database Backup and Recovery User's Guide* for more information about how to invoke RMAN. Refer to the *Oracle Database Backup and Recovery Reference* for a complete list of RMAN commands and their options.

Using SQL in RMAN

From the RMAN command line:

- Execute SQL commands and PL/SQL procedures
- Use the optional `SQL` prefix to avoid ambiguity

```
RMAN> SELECT NAME, DBID, LOG_MODE  
2> FROM V$DATABASE;  
  
NAME          DBID LOG_MODE  
-----  
ORCL          1297344416 NOARCHIVELOG
```



Copyright © 2020, Oracle and/or its affiliates.

You can execute SQL commands and PL/SQL procedures from the RMAN command line.

The `SQL` keyword is optional, but you should use it to eliminate ambiguity, especially for a few commands that exist in both RMAN and SQL and have different uses.

The RMAN `DESCRIBE` command provides the functionality of the SQL*Plus `DESCRIBE` command. You can use the abbreviated version `DESC` or the spelled-out `DESCRIBE` to list the column definitions of a table or view. To access a table or view in another schema, you must have `SELECT` privileges on the object or connect in the `SYSDBA` mode.

Types of RMAN Commands

RMAN commands are of the following types:

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

Some commands can be executed as both types.



Copyright © 2020, Oracle and/or its affiliates.

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

Stand-alone commands are executed at the RMAN prompt and are generally self-contained.

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

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

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

Job Commands: Example

Job commands appear inside a RUN command block:

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

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

Deallocated after the RUN block completes

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

- ALLOCATE CHANNEL
- SWITCH

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

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

Configuring Persistent Settings for RMAN

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



Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

Viewing Persistent Settings

To examine the persistent RMAN settings for a database:

- Use the RMAN SHOW ALL command to view all configuration settings
- Query the V\$RMAN_CONFIGURATION view to display configuration settings that have been explicitly set



Copyright © 2020, Oracle and/or its affiliates.

You can view all RMAN persistent settings by connecting to the target and using the RMAN SHOW ALL command. You can query the V\$RMAN_CONFIGURATION view in the target database to display configuration settings that have been explicitly set. In the multitenant DB, each entry in V\$RMAN_CONFIGURATION has a con_id value: 0 means the configuration applies to the entire CDB, 1 means it applies to the root container only, and any other value means that it applies to the container ID of the PDB only.

Managing Persistent Settings

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

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

- Use the SHOW command to list current settings:

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

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

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



Copyright © 2020, Oracle and/or its affiliates.

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

Specify the parallelism to be used on the device by using the PARALLELISM clause, as shown in the following:

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

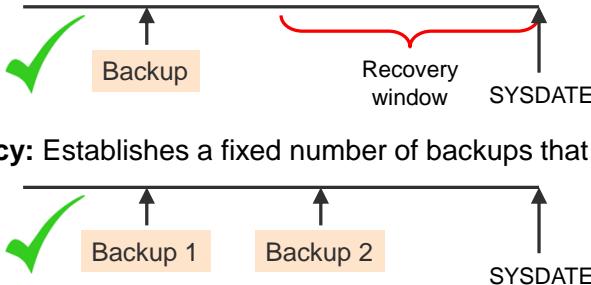
where <n> is the parallelism value.

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

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

Specifying a Retention Policy

- A retention policy describes which backups will be kept and for how long.
- There are two types of retention policies:
 - **Recovery window:** Establishes a period of time within which point-in-time recovery must be possible
 - **Redundancy:** Establishes a fixed number of backups that must be kept
- Retention policies are mutually exclusive.



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

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

Recovery Window Retention Policy

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

```
SYSDATE - backup_checkpoint_time >= recovery_window
```

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

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

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

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

- Create a backup. In this case, the synchronization is done implicitly.
- Execute the `RESYNC CATALOG` command.

Redundancy Retention Policy

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

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

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

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

Disabling the Retention Policy

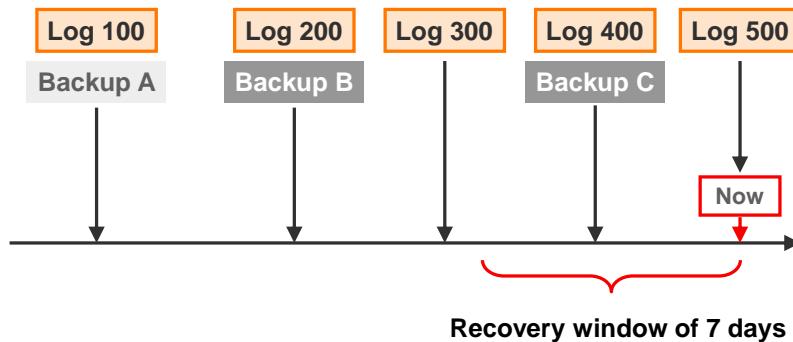
You may want to disable the retention policy totally. If you have a separate system, outside of RMAN, which backs up your disk backups to tape, you may want to do this. If you disable the retention policy, then RMAN never considers a backup obsolete. Because RMAN does not have to decide when to remove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

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

Note: You can specify that a backup is an exception to the retention policy that you have defined by creating an archival backup.

Recovery Window Retention Policy: Example

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



Backup Obsolete

Backup Not obsolete

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

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

Summary

In this lesson, you should have learned how to:

- Describe the features and functions of Recovery Manager (RMAN)
- Configure and manage RMAN settings



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Configuring the Default Backup Destination
- Setting the Date and Time Format for RMAN
- Configuring RMAN Settings



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Backup Strategies

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- Describe RMAN backup types
- Describe Oracle backup solutions
- Describe, compare, and determine your backup strategy



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Understanding Types of Backups

You can understand different types of backups by becoming familiar with these concepts:

- Backup terminology
- Types of backups
- RMAN backup types

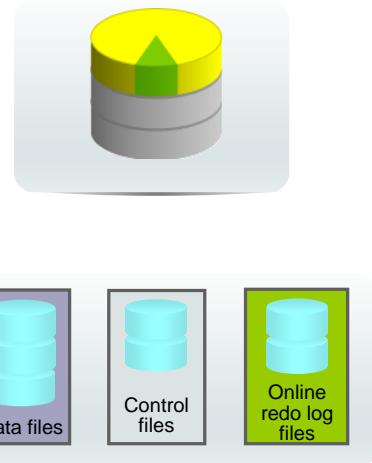


Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Backup Terminology

- *Backup strategy* may include:
 - Entire database (whole)
 - Portion of the database (partial)
- *Backup type* may indicate inclusion of:
 - All data blocks within your chosen files (full)
 - Only information that has changed since a previous backup (incremental)
 - Cumulative (changes since last level 0)
 - Differential (changes since last incremental)
- *Backup mode* may be:
 - Offline (consistent, cold)
 - Online (inconsistent, hot)



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Whole database backup: Includes all data files and at least one control file (remember that all control files in a database are identical)

Partial database backup: May include zero or more tablespaces and zero or more data files; may or may not include a control file

Full backup: Makes a copy of each data block that contains data and that is within the files being backed up

Incremental backup: Makes a copy of all data blocks that have changed since a previous backup. The Oracle database supports two levels of incremental backup (0 and 1). A level 1 incremental backup can be one of two types: *cumulative* or *differential*. A cumulative backup backs up all changes since the last level 0 backup. A differential backup backs up all changes since the last incremental backup (which could be either a level 0 or level 1 backup). Change Tracking with RMAN supports incremental backups.

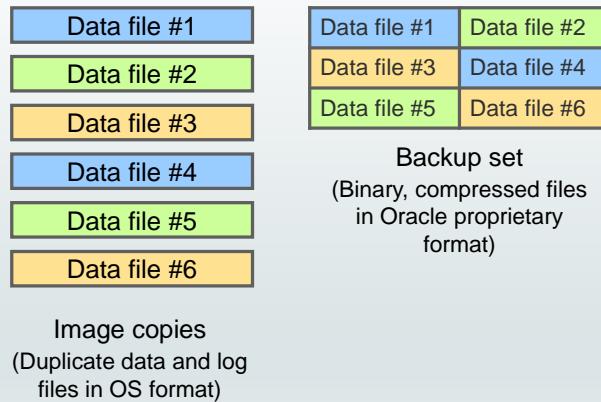
Offline backups (also known as “cold” or *consistent* backup): Are taken while the database is not open. They are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

Online backups (also known as “hot” or *inconsistent* backup): Are taken while the database is open. They are inconsistent because, with the database open, there is no guarantee that the data files are synchronized with the control files.

Understanding Types of Backups

Backups may be stored as:

- Image copies
- Backup sets
- Proxy copies



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Image copies: Are duplicates of data or archived log files (similar to simply copying the files by using operating system commands)

Backup sets: Are collections of one or more binary files that contain one or more data files, control files, server parameter files, or archived log files. With backup sets, empty data blocks are not stored, thereby causing backup sets to use less space on the disk or tape. Backup sets can be compressed to further reduce the space requirements of the backup.

Image copies must be backed up to the disk. Backup sets can be sent to the disk or directly to the tape.

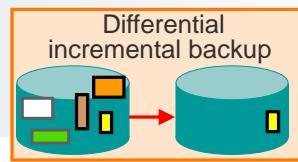
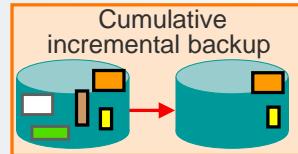
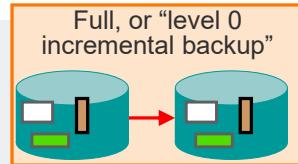
The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy, only the file or files need to be retrieved from your backup location. With backup sets, the entire backup set must be retrieved from your backup location before you extract the file or files that are needed.

The advantage of creating backups as backup sets is better space usage. In most databases, 20% or more of the data blocks are empty blocks. Image copies back up every data block, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems, the advantages of backup sets outweigh the advantages of image copies.

Proxy copies: RMAN also supports proxy copy, a feature that enables a media manager to manage completely the transfer of data between disk and backup media. RMAN supplies a list of files that need to be backed up or restored. The media manager then determines how and when to move the data. Additional information on proxy copies is provided in another lesson.

RMAN Backup Types

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



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Full Backups

A full backup is different from a whole database backup. A full data file backup is a backup that includes every used data block in the file. RMAN copies all blocks into the backup set or image copy, skipping only those data file blocks that are not part of an existing segment. For a full image copy, the entire file contents are reproduced exactly. A full backup cannot be part of an incremental backup strategy; it cannot be the parent for a subsequent incremental backup.

Incremental Backups

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

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

RMAN can create multilevel incremental backups as follows:

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

Examples

- To perform an incremental backup at level 0, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 0 DATABASE;
```
- To perform a differential incremental backup, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 DATABASE;
```
- To perform a cumulative incremental backup, use the following command:

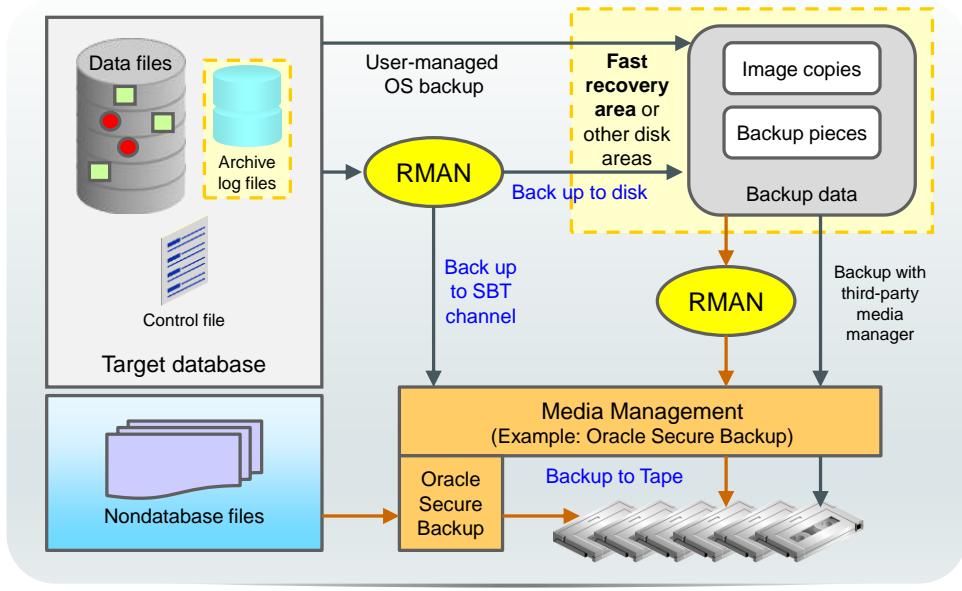
```
RMAN> BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
```

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

A full backup has no effect on subsequent incremental backups and is not considered part of any incremental backup strategy, although a full image copy backup can be incrementally updated by applying incremental backups with the RECOVER command.

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

Backup Solutions: Overview



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Recovery Manager (RMAN) is the recommended method of backing up your Oracle database. You can use it to back up to disk or to a System Backup to Tape (SBT) channel. Oracle recommends that disk backups are stored in the Fast Recovery Area (FRA).

Oracle Secure Backup complements existing functionality by adding backup to tape and backup of file system data. It interacts transparently with RMAN. Third-party media managers can also be used to back up to tape.

User-managed backups are non-RMAN backups, for example, using an OS utility. They are often based on scripts that a DBA must write.

Balancing Backup and Restore Requirements

Consideration	Performance Effect
Incremental Backup Strategy	<ul style="list-style-type: none">Improves backup performance, with trade-off in recovery performanceBlock change tracking for fast incremental backupsCumulative and differential incremental backups“Incremental forever” requires an initial full backup.
Multiplexing	<ul style="list-style-type: none">Back up files in parallel per channel to improve performance.Multiplexing level = min (FILESPERSET, MAXOPENFILES). Set MAXOPENFILES = 1 for SAME or ASM data files.Set # of RMAN channels = # of tape drives.
Hardware/Network/Storage	<ul style="list-style-type: none">Assesses host resources, production disk I/O, HBA/network, tape drive throughput.Minimum performing component of these will be a performance bottleneck.



Copyright © 2020, Oracle and/or its affiliates.

Incremental Backup Strategy

Incremental backups generally take up less space than a full backup and typically take less time to create. Recovery with incremental backups is generally faster than using redo log files to apply changes to a backup.

By incrementally updating backups, you can avoid the overhead of making full image copy backups of data files, while also minimizing time required for media recovery of your database.

The block change tracking feature for incremental backups can improve backup performance by recording changed chunks of blocks for each data file. If block change tracking is enabled on a primary or standby database, RMAN uses a block change tracking file to identify a chunk of changed blocks for incremental backups. By reading this small bitmap file, RMAN avoids having to scan every block in the data file that it is backing up.

In a *differential incremental backup*, all blocks that are changed after the most recent incremental level 1 or 0 backup are backed up. In a *cumulative incremental backup*, all blocks that are changed after the most recent level 0 incremental backup are backed up. When recovery time is more important than disk space, cumulative backups are preferable because fewer incremental backups need to be applied during recovery.

RMAN Multiplexing

An RMAN channel represents a single backup file stream. A channel can read multiple data files or archived logs into a multiplexed backup set or can read one file at a time for an image copy backup. Increasing the number of RMAN channels increases backup parallelism, which may reduce the time required to create the backup.

RMAN multiplexing refers to the number of data files or archived logs that are read by one channel at any time. The default is min (FILESPERSET, MAXOPENFILES). FILESPERSET is specified in the BACKUP command and defaults to 64. MAXOPENFILES is specified in the CONFIGURE CHANNEL command and defaults to 8. Multiplexing therefore defaults to 8, meaning that a maximum of eight files will be read by one channel at any one time.

For Stripe and Mirror Everything (SAME) and ASM storage, MAXOPENFILES should be set to 1 because all files are striped appropriately across available disks and will be read efficiently by RMAN.

Do not use media management multiplexing (multiple channels per tape drive). RMAN backup pieces will not be efficiently restored due to the interleaving of pieces on the same tape volume, which may necessitate the forward and backward movement of the tape.

Comparing Backup Strategies

Strategy	Backup Factors	Recovery Factors
Option 1: Full and Incremental Backups	Fast incremental backups Save space with backup compression. Cost-effective tape storage	Full backup restored first and then incremental backups and archived logs Tape backups read sequentially
Option 2: Incrementally Updated Disk Backups	Incremental + roll forward to create up-to-date copy Requires 1x production storage for copy	Backups read via random access Restore-free recovery with <code>SWITCH</code> command
Option 3: Offload Backups to Physical Standby Database	Above benefits + primary database free to handle more workloads Requires 1X production hardware and storage for standby database	Fast failover to standby database in the event of any failure Backups are last resort, in the event of double site failure



Copyright © 2020, Oracle and/or its affiliates.

On the next few pages, the backup strategy options presented in the table are discussed in detail.

The options provide the following major advantages:

- **Option 1:** Space and cost-effectiveness
- **Option 2:** Recovery effectiveness
- **Option 3:** Benefits of options 1 and 2 and the ability to offload production work

These examples should help you to develop your own backup and recovery strategy, which can be a combination of these options, customized to your recovery requirements and available infrastructure (tapes, disks, and so on).

Option 1: Full and Incremental Backups

- Well suited for:
 - Databases that can tolerate hours/days RTO
 - Environments where disk is premium
 - Low-medium change frequency between backups (<20%)
- Backup strategy:
 - Weekly level 0 and daily differential incremental backup sets to tape, with optional backup compression
 - Enable block change tracking so that only changed block chunks are read and written during incremental backup.
 - Back up archived logs and retain on-disk, as needed.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

In this option, you take weekly level 0 incremental backups to tape and daily differential incremental backups.

You can also take advantage of RMAN support for binary compression of backup sets. However, if your tape device performs its own compression, you should not use both RMAN backup set compression and the media manager vendor's compression. In most cases, the media manager compression provides better results.

When you enable block change tracking, RMAN uses a block change tracking file to identify modified block chunks, and backs up only those blocks. Instead of scanning every block in the data file that is being backed up, RMAN reads the small bitmap file to determine which blocks have changed.

Option 2: Incrementally Updated Disk Backups

- Well suited for:
 - Databases that can tolerate no more than a few hours RTO
 - Environments where disk can be allocated for 1x size of database or most critical tablespaces
- Backup strategy:
 - Initial image copy to FRA, daily incremental backups
 - New on-disk copy by using incrementals to roll forward copy
 - Full backup archived to tape as needed
 - Archived logs backed up and retained on-disk as needed
 - Fast recovery from disk or `SWITCH` to use image copies



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

In this option, you create incrementally updated backups. Create an image copy of each data file and then apply daily level 1 incremental backups to roll forward the image copy. With this technique, you avoid the overhead of creating multiple full image copies of your data files, and you can take advantage of using image copies during a recovery operation to minimize recovery time.

Option 3: Offloading Backups to Physical Standby Database in Data Guard Environment

- Well suited for:
 - Databases that require no more than several minutes of recovery time in the event of any failure
 - Environments that can preferably allocate symmetric hardware and storage for physical standby database
 - Environments with tape infrastructure that can be shared between primary and standby database sites
- Backup strategy:
 - Full and incremental backups offloaded to physical standby database
 - Fast incremental backup on standby with Active Data Guard
 - Backups restored to primary or standby database
 - Backups taken at each database for optimal local protection



Copyright © 2020, Oracle and/or its affiliates.

In a Data Guard environment, you can offload incremental backups to a physical standby database. Incremental backups of a standby and primary database are interchangeable. You can apply an incremental backup of a standby database to a primary database or apply an incremental backup of a primary database to a standby database.

Note: The Oracle Active Data Guard option is required for the use of block change tracking on the physical standby database.

Backing Up Read-Only Tablespaces: Considerations

- Backup optimization causes RMAN to back up read-only tablespaces only when no backup exists that satisfies the retention policy.
- If you change the tablespace to read/write, back it up immediately.
- You can use the `SKIP READONLY` option of the RMAN `BACKUP` command to skip read-only tablespaces or data files.



Copyright © 2020, Oracle and/or its affiliates.

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

Data Warehouse Backup and Recovery: Best Practices

- Exploit partitioning and read-only tablespaces:
 - Older partitions that contain data that is no longer changing can be moved to read-only tablespaces.
 - Back up read-only tablespaces once and then periodically depending on tape retention policy.
- Divide full backup workload across multiple days.
- Leverage database and backup compression.
- Save time with tablespace-level backups.
 - Back up index tablespaces less frequently than data tablespaces.
 - Back up scarcely used tablespaces less frequently.
 - Reduce restore time for most critical tablespaces, by grouping them together in separate backup sets.
- Take incremental backup when NOLOGGING operations finish to ensure recoverability.



Copyright © 2020, Oracle and/or its affiliates.

By using a command similar to the following, you can divide the full backup workload across multiple days:

```
BACKUP DATABASE NOT BACKED UP SINCE 'SYSDATE-3' DURATION 06:00 PARTIAL  
MINIMIZE TIME;
```

When you execute the command on the first day, a full backup begins. It executes for six hours. On the next day when the command executes again, the backup starts with the last file not backed up and resumes the full backup, again in a six-hour backup window. Over the course of a few days, the entire database will be backed up.

Summary

In this lesson, you should have learned how to:

- Describe RMAN backup types
- Describe Oracle backup solutions
- Describe, compare, and determine your backup strategy



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Creating Database Backups



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

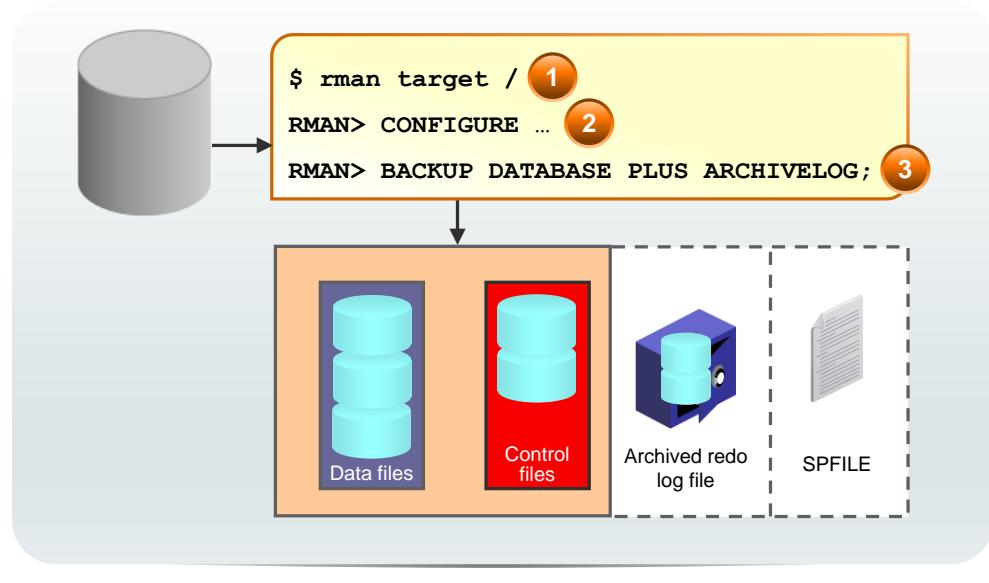
- Create whole backups
- Create full and incremental backups
- Configure block change tracking
- Use Oracle-suggested backup strategy
- Back up the control file to a trace file
- Report and manage backups

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Using RMAN Commands to Create Backups



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

1. In a terminal session, start RMAN and connect to the target database.
2. Execute configuration commands:
 - CONFIGURE DEFAULT DEVICE TYPE TO disk;
 - CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY;
3. A whole database backup is a copy of all data files and the control file. You can optionally include the server parameter file (SPFILE) and archived redo log files. Using RMAN to make an image copy of all the database files simply requires mounting or opening the database, starting RMAN, and entering the BACKUP command shown in the slide.

Optionally, you can supply the `DELETE INPUT` option when backing up archive log files. That causes RMAN to remove the archive log files after backing them up. This is useful especially if you are not using a fast recovery area, which would perform space management for you, deleting files when space pressure grows. In that case, the command in the slide would look like the following:

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

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

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

Syntax and Clauses in RMAN

```
$ export ORACLE_SID=cdb1
$ rman TARGET /
```

- DATABASE keyword operates on all PDBs and CDB root or on only one PDB.

```
RMAN> BACKUP DATABASE;
RMAN> RECOVER DATABASE;
```

- PLUGGABLE DATABASE keywords operate on individual PDBs.

```
RMAN> BACKUP PLUGGABLE DATABASE hr_pdb, sales_pdb;
RMAN> RECOVER PLUGGABLE DATABASE hr_pdb;
```

- Back up, restore, recover the CDB root using CDB\$ROOT keyword.

```
RMAN> BACKUP PLUGGABLE DATABASE "CDB$ROOT";
```

- Qualify tablespace of PDB with PDB name.

```
RMAN> BACKUP TABLESPACE sales_pdb:tbs2;
RMAN> RESTORE TABLESPACE system;
```



Copyright © 2020, Oracle and/or its affiliates.

You can use Recovery Manager (RMAN) or Enterprise Manager to back up and recover entire CDBs, partial CDBs, individual whole PDBs, or partial PDBs such as tablespace/datafile of specific PDBs.

The CDB or PDBs are the possible targets for RMAN; an individual PDB is a valid RMAN TARGET database. Connect to the CDB root or a PDB as a user with SYSDBA or SYSBACKUP privilege .

The traditional syntax, such as BACKUP DATABASE, RESTORE DATABASE, and RECOVER DATABASE, operates on the CDB root and all its PDBs and, therefore, on the whole CDB or on a single PDB depending on the connection.

- If you are connected to the CDB root, the BACKUP DATABASE backs up the CDB root and all its PDBs, including the controlfile and the server parameter file (SPFILE).
- If you are connected to a PDB, the BACKUP DATABASE backs up the PDB datafiles.

The PLUGGABLE DATABASE syntax allows backup of, restore, and recover single or several PDBs.

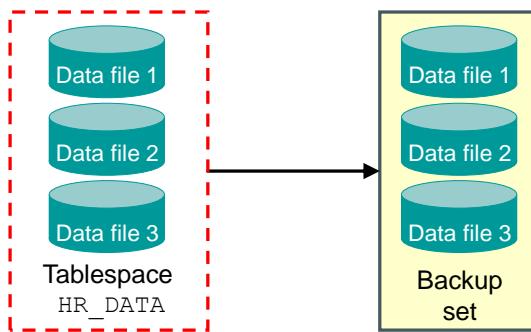
To back up, restore, or recover, the CDB root, CDB\$ROOT, must be used and therefore must be at least mounted.

The RMAN TABLESPACE syntax can be qualified with the PDB name so that a user can specify the name of the tablespace as it is known to PDB during backup, restore, and recover commands. If no PDB qualifier is used, then the CDB root is used by default. To list the tablespaces and their associated PDB, use the REPORT SCHEMA syntax.

The same new clauses can be applied to DUPLICATE, SWITCH, REPORT, CONVERT, CHANGE, LIST, DELETE.

Creating Backup Sets

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



ORACLE®

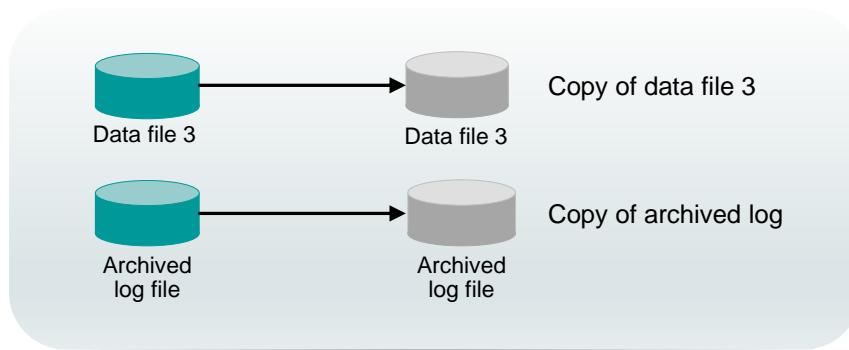
Copyright © 2020, Oracle and/or its affiliates.

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

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

Creating Image Copies

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



ORACLE®

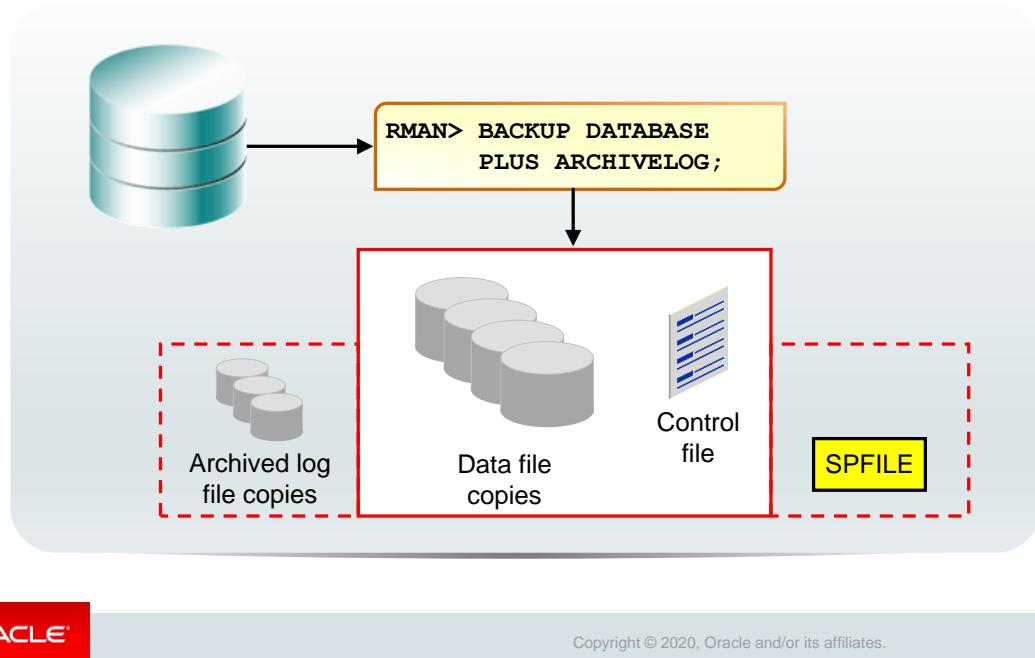
Copyright © 2020, Oracle and/or its affiliates.

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

An image copy has the following characteristics:

- An image copy can be written only to disk. When large files are being considered, copying may take a long time, but restoration time is reduced considerably because the copy is available on the disk.
- If files are stored on disk, they can be used immediately by using the `SWITCH` command in RMAN, which is equivalent to the `ALTER DATABASE RENAME FILE` SQL statement.
- In an image copy, all blocks are copied, whether they contain data or not, because an Oracle database process copies the file and performs additional actions such as checking for corrupt blocks and registering the copy in the control file.
- To speed up the process of copying, you can use the `NOCHECKSUM` parameter. By default, RMAN computes a checksum for each block backed up and stores it with the backup. When the backup is restored, the checksum is verified.
- An image copy can be part of a full or incremental level 0 backup because a file copy always includes all blocks. You must use the level 0 option if the copy will be used in conjunction with an incremental backup set.

Creating a Whole Database Backup



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

A whole database backup can be made using either backup sets or image copies of the entire set of data files and must include the control file (included automatically). You can optionally add the server parameter file (**SPFILE**) and archived redo log files. Using Recovery Manager (RMAN) to make an image copy of all the database files simply requires mounting or opening the database, starting RMAN, and entering the **BACKUP** command shown in the slide.

Optionally, you can supply the **DELETE INPUT** option when backing up archive log files. That causes RMAN to remove the archive log files after backing them up. This is useful especially if you are not using a fast recovery area, which would perform space management for you, deleting files when space pressure grows. An example follows:

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

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

- **CONFIGURE DEFAULT DEVICE TYPE TO disk;**
- **CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY;**

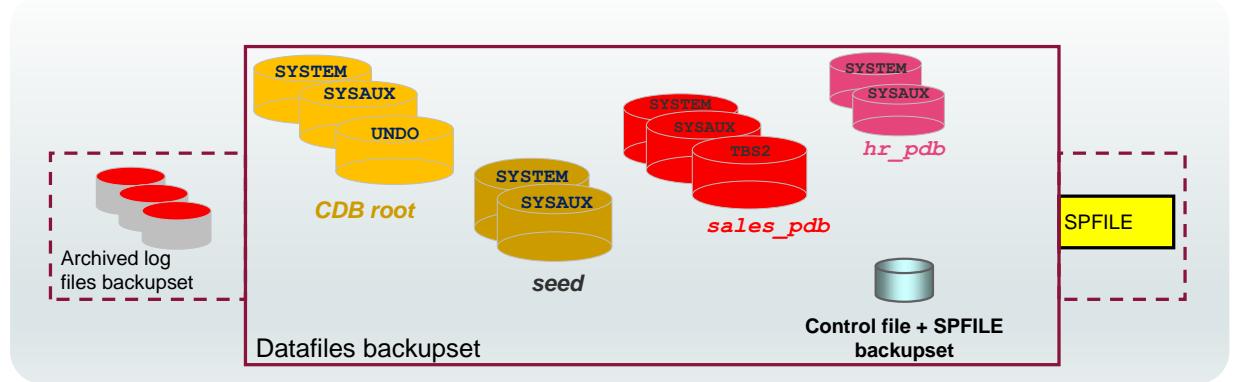
To make a separate backup of **SPFILE** means to export it to a text file with the following SQL command:

- **CREATE PFILE FROM SPFILE;**

CDB Backup: Whole CDB Backup

Back up all PDBs datafiles and CDB root files.

```
$ rman TARGET /
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

RMAN can back up the entire CDB and contained or individual PDBs. In addition, individual tablespaces or datafiles can be backed up from a specific PDB.

A whole database backup of a CDB can be, in a similar way to non-CDBs, either backup sets or image copies of the entire set of datafiles, namely, the root datafiles and all PDBs datafiles, and the control file. You can optionally include the SPFILE and archived redo log files.

Using RMAN to make an image copy of all the CDB files simply requires mounting or opening the CDB, starting RMAN, connecting to the root with `SYSDBA` or `SYSBACKUP` privilege, and entering the `BACKUP` command shown in the slide. Optionally, you can supply the `DELETE INPUT` option when backing up archivelog files.

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

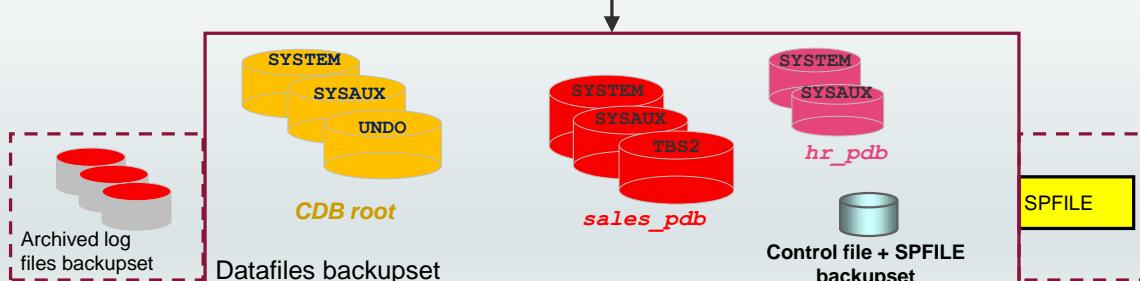
```
RMAN> BACKUP COPY OF DATABASE;
```

CDB Backup: Partial CDB Backup

Back up the CDB root and/or individual PDBs.

```
$ rman TARGET /
RMAN> BACKUP PLUGGABLE DATABASE "CDB$ROOT", sales_pdb;
RMAN> BACKUP PLUGGABLE DATABASE hr_pdb PLUS ARCHIVELOG;
```

```
$ rman TARGET sys@hr_pdb
RMAN> BACKUP DATABASE;
```



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

A partial CDB backup backs up the entire set of datafiles of the CDB root, all datafiles of defined PDBs, and the control file and SPFILE because it has been configured to be backed up automatically after each backup.

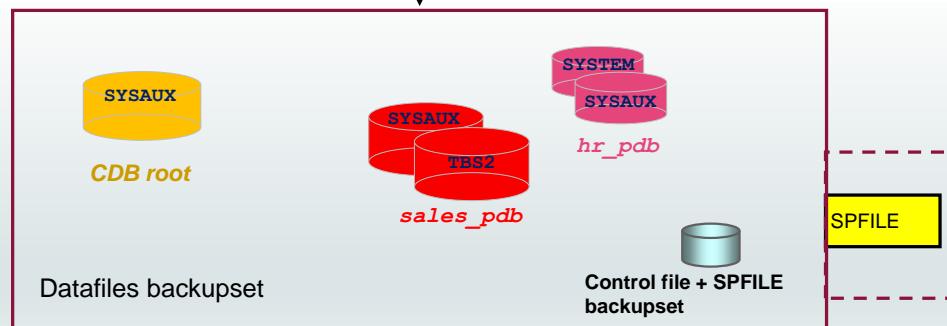
The command `BACKUP PDB "CDB$ROOT", sales_pdb`, backs up all datafiles of the root container, namely, the `SYSTEM`, `SYSAUX`, and `UNDO` datafiles and then all datafiles of the `sales_pdb` PDB, namely, the `SYSTEM`, `SYSAUX`, and `TBS2` datafiles.

If you connect to the PDB with RMAN like in the second example, some options become invalid such as `PLUS ARCHIVELOG`.

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
...
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6587334BFE4E3465E05321339
60A012c/backupset/2018_02_23/o1_mf_nnndf_TAG20180223T021454_f8yy8z9n_.bk
p tag=TAG20180223T021454 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:16
Finished backup at 23-FEB-18
Starting backup at 23-FEB-18
using channel ORA_DISK_1
skipping archived logs when connected to a PDB
backup cancelled because there are no files to backup
Finished backup at 23-FEB-18
```

PDB Backup: Partial PDB Backup

```
$ rman TARGET /
RMAN> REPORT SCHEMA;
RMAN> BACKUP TABLESPACE sales_pdb:tbs2;
RMAN> BACKUP TABLESPACE hr_pdb:system, sales_pdb:sysaux;
RMAN> BACKUP TABLESPACE sysaux, hr_pdb:sysaux;
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

A partial PDB backup backs up the datafiles named tablespaces of individual PDBs and the control file and SPFILE because it has been configured to be backed up automatically at each backup performed.

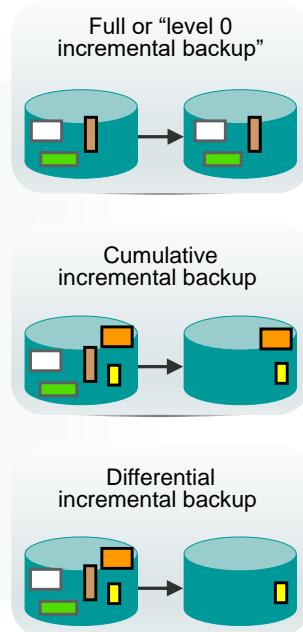
The example uses the command `BACKUP TABLESPACE` to back up all datafiles of tablespace TBS2 of `sales_pdb` PDB. To find the names of tablespaces within PDBs, use the `REPORT SCHEMA` command.

The second backup uses the same command to back up all datafiles of tablespace SYSTEM of `hr_pdb` PDB and all datafiles of tablespace SYSAUX of `sales_pdb` PDB.

The third backup uses the same command to back up all datafiles of tablespace SYSAUX of the CDB root and all datafiles of tablespace SYSAUX of `hr_pdb` PDB.

Review: RMAN Backup Types

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



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Full Backups

A full backup is different from a whole database backup. A full data file backup is a backup that includes every used data block in the file. RMAN copies all blocks into the backup set or image copy, skipping only those data file blocks that are not part of an existing segment. For a full image copy, the entire file contents are reproduced exactly. A full backup cannot be part of an incremental backup strategy; it cannot be the parent for a subsequent incremental backup.

Incremental Backups

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

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

RMAN can create multilevel incremental backups as follows:

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

Examples

- To perform an incremental backup at level 0, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

- To perform a differential incremental backup, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 DATABASE;
```

- To perform a cumulative incremental backup, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
```

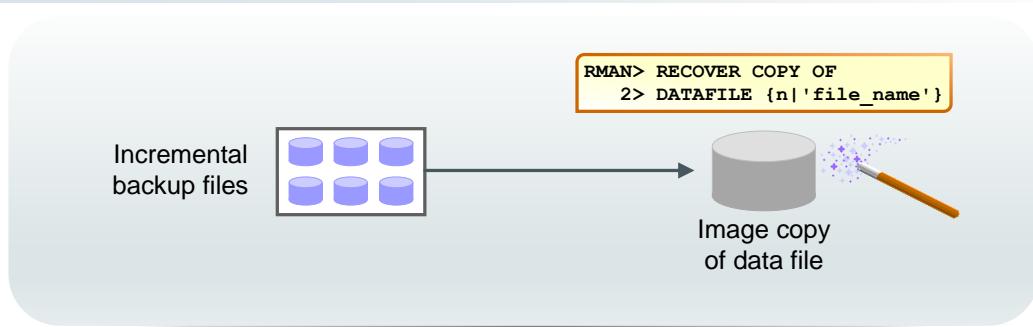
RMAN makes full backups by default if neither FULL nor INCREMENTAL is specified. Unused block compression causes never-written blocks to be skipped when backing up data files to backup sets, even for full backups.

A full backup has no effect on subsequent incremental backups and is not considered part of any incremental backup strategy, although a full image copy backup can be incrementally updated by applying incremental backups with the RECOVER command.

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

Incrementally Updated Backups

- Image copies are updated with all changes up to the incremental backup SCN.
- Incremental backup reduces the time required for media recovery.
- With incrementally updated backups, you can use the `SWITCH` command during the recovery operation.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

You can use RMAN to apply incremental backups to data file image copies.

```
BACKUP AS COPY INCREMENTAL LEVEL 0 DATABASE;
```

With the use of incrementally updated backups, you can use the `SWITCH` command during the recovery operation.

RMAN can roll forward (recover) an image copy to the specified point in time by applying the incremental backups to the image copy. The image copy is updated with all changes up through the SCN at which the incremental backup was taken. RMAN uses the resulting updated data file in media recovery just as it would use a full image copy taken at that SCN, without the overhead of performing a full image copy of the database every day. The following are the benefits of applying incremental backups to data file image copies:

- You reduce the time required for media recovery (using archive logs) because you need to apply archive logs only since the last incremental backup.
- You do not need to perform a full image copy after the incremental restoration.

If the recovery process fails during the application of the incremental backup file, you simply restart the recovery process. RMAN automatically determines the required incremental backup files to apply, from before the image data file copy until the time at which you want to stop the recovery process. If there is more than one version of an image copy recorded in the RMAN catalog, RMAN automatically uses the latest version of the image copy. RMAN reports an error if it cannot merge an incremental backup file with an image copy.

Incrementally Updated Backups: Example

If you execute these commands daily:

```
RMAN> recover copy of database with tag 'daily_inc';
RMAN> backup incremental level 1 for recover of copy
2> with tag 'daily_inc' database;
```

This is the result:

	RECOVER	BACKUP
Day 1	Nothing	Create image copies
Day 2	Nothing	Create incremental level 1
Day 3 and onward	Recover copies based on incremental	Create incremental level 1



Copyright © 2020, Oracle and/or its affiliates.

If you execute the commands shown in the slide daily, you get continuously updated image copies of all the database data files at any time.

The chart shows what happens for each run. Note that this algorithm requires some priming; the strategy does not come to fruition until after day 3.

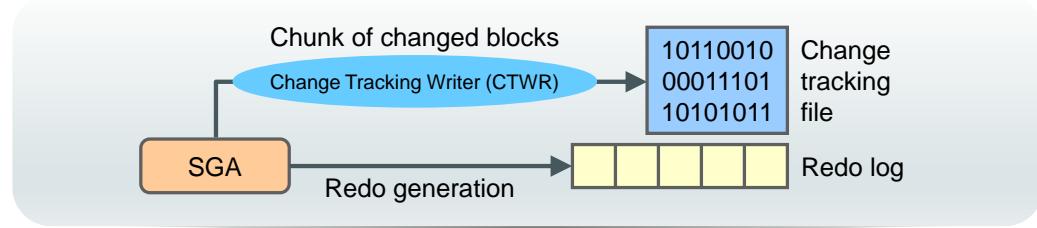
- **Day 1:** The RECOVER command does nothing. There are no image copies to recover. The BACKUP command creates the image copies.
- **Day 2:** The RECOVER command, again, does nothing. This is because there is no incremental backup yet. The BACKUP command creates the incremental backup, now that baseline image copies have been created on day 1.
- **Day 3:** The RECOVER command applies the changes from the incremental backup to the image copies. The BACKUP command takes another incremental backup, which will be used to recover the image copies on day 4. The cycle continues like this.

It is important to use tags when implementing this kind of backup strategy. They serve to link these particular incremental backups to the image copies that are made. Without the tag, the most recent, and possibly incorrect, incremental backup would be used to recover the image copies.

Fast Incremental Backup

Implemented by block change tracking, which:

- Maintains a record of block chunks that have changed since the last backup
- Writes this record to a file, as redo is generated
- Is automatically accessed when a backup is done and can make the backup complete more quickly
- Is optimized for up to eight incremental backups
- Is recommended if the changes are less than 20 percent



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

You can perform fast incremental backup by enabling block change tracking (BCT). The BCT file can contain only eight bitmaps, so the backup cannot be optimized if there have been more than eight incremental backups since the parent-level backup that the new incremental will be based on.

Consider the eight-bitmap limit when developing your incremental backup strategy. For example, if you make a level 0 database backup followed by seven differential incremental backups, then the block change tracking file now includes eight bitmaps. If you then make a cumulative level 1 incremental backup, RMAN cannot optimize the backup because the bitmap corresponding to the parent level 0 backup is overwritten with the bitmap that tracks the current changes.

When it is time to perform the incremental backup, RMAN can look at the block change tracking file, determine the modified block chunks, and back up only those blocks. It does not have to scan every block to see whether it has changed since the last backup. This can make incremental backup faster.

Block change tracking is recommended when your changes are 20 percent or less.

The maintenance of the tracking file is fully automatic and does not require your intervention. The minimum size for the block change tracking file is 10 MB, and any new space is allocated in 10 MB increments. Oracle Database server does not record block change information by default.

Maintaining the Block Change Tracking File

- The DB_CREATE_FILE_DEST initialization parameter provides the default destination.
- Enable or disable with:

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

- Rename the block change tracking file with the ALTER DATABASE RENAME command. (The database must be in MOUNT state.)



Copyright © 2020, Oracle and/or its affiliates.

You can enable block change tracking in multiple tools.

For example, in Enterprise Manager Cloud Control, navigate to the database instance home page > Availability > Backup & Recovery > Backup Settings > Policy. You do not need to set the block change tracking file destination if the DB_CREATE_FILE_DEST initialization parameter is set. But you can specify the name of the block change tracking file, placing it in any location you choose.

You can also enable or disable this feature by using an ALTER DATABASE command. If the change tracking file is stored in the database area with your database files, it is deleted when you disable change tracking.

You can rename the block change tracking file by using the ALTER DATABASE RENAME command. Your database must be in the MOUNT state to rename the tracking file. The ALTER DATABASE RENAME FILE command updates the control file to refer to the new location. You can use the following syntax to change the location of the block change tracking file:

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

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

Monitoring Block Change Tracking

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

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

FILE#  BLOCKS_IN_FILE  BLOCKS_READ PCT_READ_FOR_BACKUP BLOCKS_BACKED_UP
-----  -----  -----  -----
  1        56320       4480            7             462
  2        3840        2688           70            2408
  3        49920       16768           33            4457
```



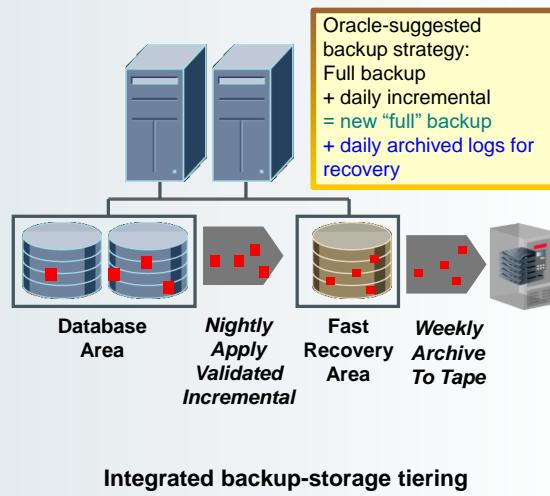
Copyright © 2020, Oracle and/or its affiliates.

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

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

Automatic Disk-to-Disk Backup and Recovery

- Integrated disk-to-disk backup and recovery:
Low-cost disks used for fast recovery area
- Fast incremental backups:
Back up only changed blocks
- Nightly incremental backup rolls forward recovery area backup:
No need to do full backups



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

RMAN fully automates disk-based backup and recovery by using essentially a tiered storage configuration (as shown in the slide). You can use different types of storage for the database files and the fast recovery area (FRA).

With the **Oracle-suggested backup strategy**, you perform a full backup of your database (an image copy of each data file). Then set up automated nightly incremental backups, stored in the FRA. Only changed data blocks are backed up, which save considerable storage space. You can implement block change tracking to track the changed blocks more efficiently. The nightly incremental backups can be used to roll forward the database backup (by applying an incremental level 1 backup) and automatically create a full backup.

This procedure enables faster backups by propagating changes to the FRA. In addition, restores are faster because backup files are copied from the FRA or a copy of the file in the FRA. (If recovery is needed, then only the daily archived redo logs need to be applied to yesterday's full backup.)

Oracle-Suggested Backup

- Provides an out-of-the-box backup strategy based on the backup destination
- Sets up recovery window for backup management
- Schedules recurring and immediate backups

Full backup
+ daily incremental
= new “full” backup
+ daily archived logs for recovery



Copyright © 2020, Oracle and/or its affiliates.

Enterprise Manager makes it easy for you to set up an Oracle-suggested backup strategy that protects your data and provides efficient recoverability. By default, there is a 24-hour recovery window from disk. The Oracle-suggested strategy uses the incremental backup and incrementally updated backup features, providing faster recoverability than is possible when applying database changes from the archived log files.

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

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

You can change this: depending on your organization’s disk capacity, three (to seven) days are often used.

Backing Up the Control File to a Trace File

- A control file trace backup contains the SQL statement required to re-create the control files in the event that all control files are lost.
- It is recommended to do after each change in the physical structure of the database.
- Control file trace backups may be used to recover from loss of all control files.
- Choose your DBA tool: EM Express, Cloud Control, or command line.



Copyright © 2020, Oracle and/or its affiliates.

Trace files are copies of the control files. A control file trace backup contains the SQL statement required to re-create the control files in the event that all control files are lost.

Although it is very unlikely that a properly configured database (with multiple copies of the control file placed on separate disks and separate controllers) would lose all control files at the same time, it is possible. Therefore, you should back up the control file to a trace file after each change to the physical structure of the database (adding tablespaces or data files).

You can perform the task in several DBA tools: Enterprise Manager Database Express, Enterprise Manager Cloud Control, or command line:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

The trace backup is created in the location specified by the `DIAGNOSTIC_DEST` initialization parameter.

Backing Up the Control File to a Trace File

- Control files can be backed up to a trace file, generating a SQL command to re-create the control file.
- Control file trace backups may be used to recover from the loss of all control files.

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE
```



Copyright © 2020, Oracle and/or its affiliates.

In Enterprise Manager Database Express, select Storage > Control Files to manage your database's control files. Control files have an additional backup option; they may be backed up to a trace file. A control file trace backup contains the SQL statement that is required to re-create the control files in the event that all control files are lost.

Although it is very unlikely that a properly configured database (with multiple copies of the control file placed on separate disks and separate controllers) would lose all control files at the same time, it is possible. Therefore, you should back up the control file to a trace file after each change to the physical structure of the database (adding tablespaces or data files or adding additional redo log groups).

Trace copies of the control file can be created by using Enterprise Manager Database Express, Enterprise Manager Cloud Control, or the following SQL command:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE
```

The trace backup is created in the location specified by the `DIAGNOSTIC_DEST` initialization parameter. For example, in this course, the trace file for the `orcl` database is found in the `/u01/app/oracle/diag/rdbms/orcl/orcl/trace` directory and will have a file name such as `orcl_ora_924.trc`.

Cataloging Additional Backup Files

Using the CATALOG command:

- To catalog existing backup files that are no longer listed in the control file
- To catalog files that were never included in the control file or recovery catalog
- To add the following file types to the recovery catalog:
 - CONTROLFILECOPY: Control file copies
 - DATAFILECOPY: Data file copies
 - BACKUPPIECE: Backup pieces
 - ARCHIVELOG: Archived redo log files
- With the START WITH option:

```
RMAN> CATALOG ARCHIVELOG '/disk1/arch_logs/archivel_731.log',
      '/disk1/arch_logs/archivel_732.log';
RMAN> CATALOG START WITH '/tmp/arch_logs/';
```



Copyright © 2020, Oracle and/or its affiliates.

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

Example to catalog all files in the currently enabled fast recovery area:

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

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

The second example in the slide catalogs all types of backup files that are found in the /tmp/arch_logs directory.

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

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

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

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

Reporting on Backups

RMAN commands:

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

Enterprise Manager Cloud Control:

- Graphical, customizable interface



Copyright © 2020, Oracle and/or its affiliates.

Use the RMAN `LIST` command to display information about backup sets, proxy copies, and image copies recorded in the repository. There are a variety of ways in which you can list backup information.

Use the RMAN `REPORT` command to analyze information in the RMAN repository in more detail.

The `REPORT NEED BACKUP` command is used to identify all data files that need a backup. The report assumes that the most recent backup would be used in the event of a restore.

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

Using Dynamic Views

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

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



Copyright © 2020, Oracle and/or its affiliates.

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

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

To query the RC_BACKUP_FILES view, you must first execute the following in the recovery catalog database:

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

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

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

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

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

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

Summary

In this lesson, you should have learned how to:

- Create whole backups
- Create full and incremental backups
- Configure block change tracking
- Use Oracle-suggested backup strategy
- Back up the control file to a trace file
- Report and manage backups



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Backing up the Control File
- Verifying Automatic Backups of the Control File and SPFILE
- Creating a Whole Database Backup
- Creating Partial Database Backups
- Configuring Block Change Tracking
- Using Incremental Backup
- Backing Up Additional Database Files



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

6

Using Optional Backup Features

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- Compress backups
- Create multisection backups of very large files
- Create proxy copies
- Create duplexed backup sets
- Create archival backups



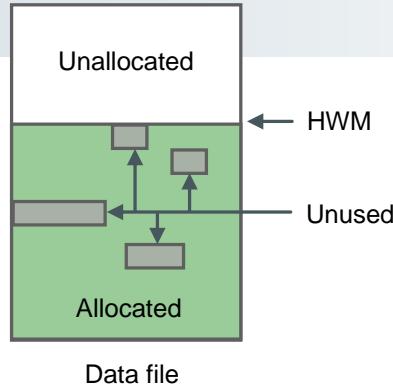
Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Saving Backup Space with Unused Block Compression

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

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



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

When certain types of backups occur, RMAN is able to skip some blocks. Unallocated blocks, which are above the high-water mark (HWM), may be skipped. Also, some allocated blocks that no longer belong to a segment (are not in use) may be skipped, provided the following are true:

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

Compressing Backups

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

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

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

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

```
BACKUP AS COMPRESSED BACKUPSET DATABASE PLUS ARCHIVELOG;
```



Copyright © 2020, Oracle and/or its affiliates.

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

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

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

You can configure compression per device type or individually for a backup set as shown in the slide.

Using RMAN Backup Compression

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



Copyright © 2020, Oracle and/or its affiliates.

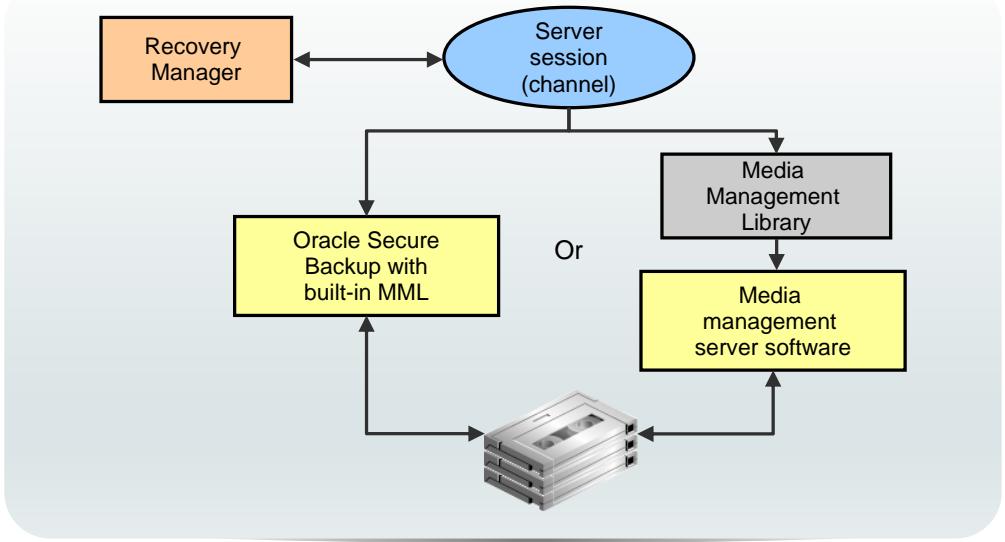
Binary compression of backup sets is supported with the algorithm settings as shown in the slide. All modes except BASIC require the Oracle Advanced Compression option.

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

The following level or compression ratios are available:

- LOW: This level is the fastest. It uses the least CPU, but provides less compression than MEDIUM.
- MEDIUM: This level provides a good balance of CPU usage and compression ratio.
- HIGH: This level provides the best compression ratio, but consumes the most CPU.
- BASIC: This offers a compression ratio comparable to MEDIUM, at the expense of additional CPU consumption.

Using a Media Manager



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

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

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

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

Backup and Restore Operations Using a Media Manager

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

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

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

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

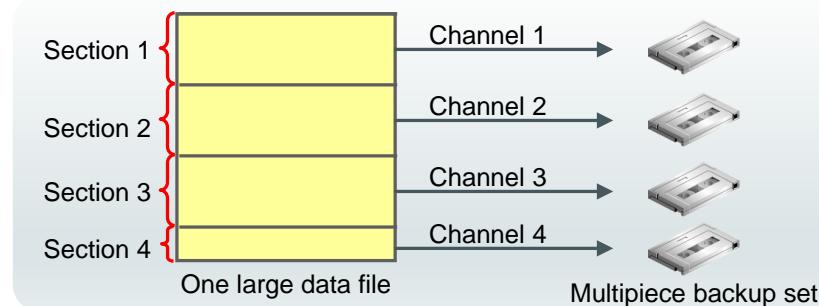
1. Oracle Database server requests the restoration of a particular file.
2. The media manager identifies the tape containing the file and reads the tape.
3. The media manager passes the information back to the Oracle database session.
4. Oracle Database server writes the file to disk.

RMAN also supports *proxy copy*, a feature that enables a media manager to manage completely the transfer of data between disk and backup media. RMAN supplies a list of files that need to be backed up or restored. The media manager then determines how and when to move the data.

Configuring Backup and Restore for Very Large Files

Multisection backups of a single file:

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



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

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

Backing Up and Restoring Very Large Files

- Multisection backups of a single data file are:
 - Created by RMAN, with your specified size value
 - For backup sets and image copies
 - For full and incremental backups
- Benefits:
 - Reduce image copy creation time
 - Are processed independently (serially or in parallel)
 - Benefit Exadata
- Do not apply large values of parallelism to back up a large file that resides on a small number of disks.



Copyright © 2020, Oracle and/or its affiliates.

How Can This Help?

Multisection image copies reduce the image copy creation time for large data files, in particular benefiting Exadata environments. This can also reduce completion time for nonbackup use cases, for example, copying a file as part of transportable tablespace procedure or creating a clone with active database duplication.

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

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

Creating RMAN Multisection Backups

RMAN command syntax:

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

Example:

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



Copyright © 2020, Oracle and/or its affiliates.

The BACKUP and VALIDATE DATAFILE commands accept the following option:

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

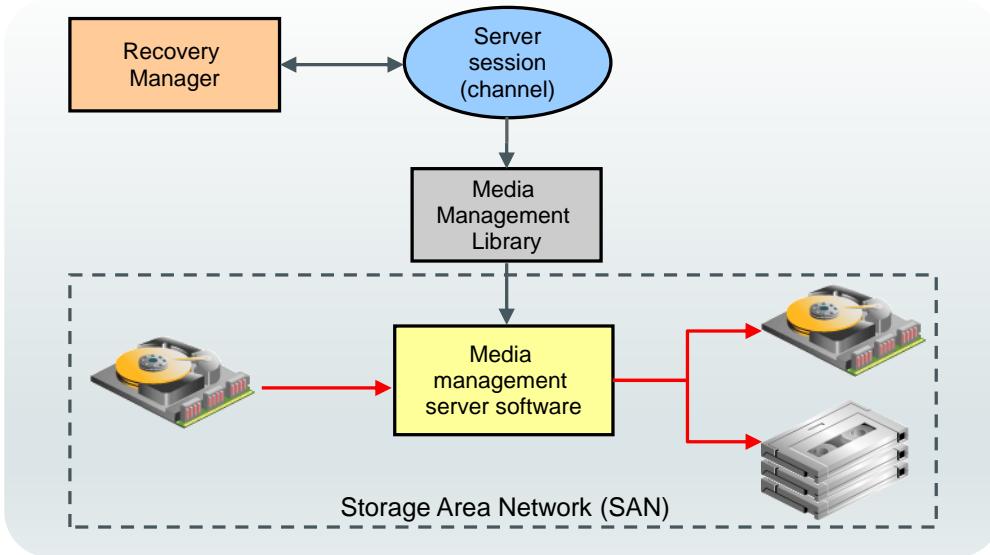
Use this to specify your planned size for each backup section. The option is both a backup command and backup spec-level option so that you can apply different section sizes to different files in the same backup job.

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

Viewing Metadata About Your Multisection Backup

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

Creating Proxy Copies



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

Syntax:

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

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

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

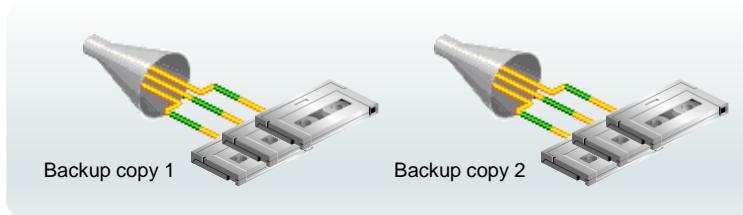
You can use the `RMAN LIST BACKUP` command to view information about the proxy copy. The `V$PROXY_DATAFILE` view contains descriptions of data file and control file proxy copy backups.

`V$PROXY_ARCHIVEDLOG`, `V$PROXY_ARCHIVEDLOG_SUMMARY`, `V$PROXY_ARCHIVEDLOG_DETAILS`, `V$PROXY_COPY_SUMMARY`, and `V$PROXY_COPY_DETAIL` also provide information about proxy copies. See *Oracle Database Reference* for a complete description of these views.

Creating Duplexed Backup Sets by Using BACKUP COPIES

Example of creating two copies of the backup set on tape:

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



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

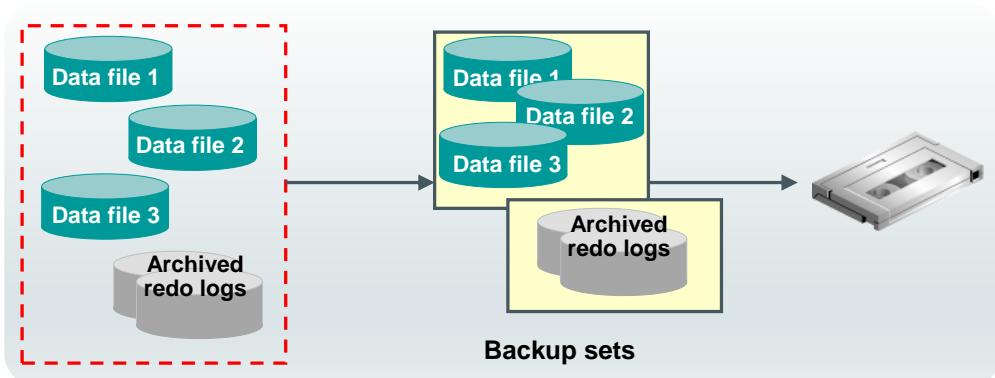
1. Specify the number of identical copies with the **COPIES** option of the **BACKUP** command.
2. Issue a **LIST BACKUP** command to verify your backup.

Duplexed backups require additional resources. For example, if a nonduplexed backup to tape uses three RMAN channels (each requiring one tape drive), then duplexing this backup (making two copies) requires six tape drives (as shown in the slide).

Underscore and number (_1, _2, and so on) are appended to the backup piece name to denote it as a duplexed copy.

Creating Backups of Backup Sets

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



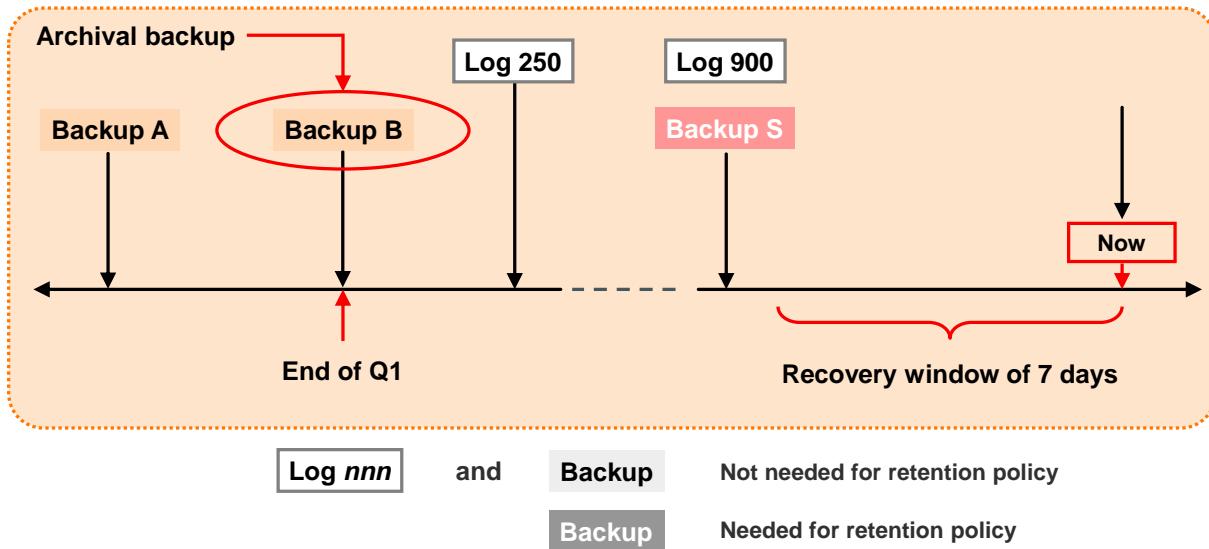
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

Archival Backups: Concepts



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

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

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

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

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

Creating Archival Backups with RMAN

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

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

- List all restore points known to the RMAN repository:

```
LIST RESTORE POINT ALL;
```

- Display a specific restore point:

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



Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

Managing Archival Database Backups

- 1 Archiving a database backup:

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

- 2 Changing the status of a database copy:

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

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

Backing Up Recovery Files

- Back up only the files in the fast recovery area:

```
RMAN> BACKUP RECOVERY AREA
```

- Back up all recovery files:

```
RMAN> BACKUP RECOVERY FILES
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

You cannot specify `DEVICE TYPE DISK` for either of these commands. To back up to disk, you must specify a directory or an Automatic Storage Management (ASM) disk group in the `TO DESTINATION` subclause.

Note: RMAN backs up only database files: data files, control files, SPFILEs, archive log files, and backups of these files. Placing an operating system file in the fast recovery area does not cause it to be included with a backup of the recovery area.

Summary

In this lesson, you should have learned how to:

- Compress backups
- Create multisection backups of very large files
- Create proxy copies
- Create duplexed backup sets
- Create archival backups



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Creating an Archival Backup



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Tuning RMAN Backup Performance

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- Interpret the RMAN message output
- Apply best-practice tuning principles
- Diagnose RMAN performance issues



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Is There a Problem?

- Know the performance of each of your components.
- Analyze the read and process steps by using the `BACKUP VALIDATE` command.
- Analyze the read and process steps by using the `RESTORE VALIDATE` command.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The first step in tuning is analysis. Do you actually have a problem that can be addressed by tuning RMAN operations? To find out:

- You need to know the performance of each of your components. For example, if you think of the speed of a tape drive, you compare the actual data rate of a backup with the documented maximum speed of that tape drive.
- For backups, analyze the read and process steps with the `BACKUP VALIDATE` command. It causes RMAN to perform all steps of the backup up to the point of writing the data to the output device; then the data is discarded, and nothing is written. Note that `VALIDATE` does perform compression if specified on the backup command, but not encryption. If compression was specified on the backup command, then `BACKUP VALIDATE` should be first run without compression, to measure the speed at which RMAN can read the input files, and then again with compression to see the effects of the compression itself.
- The `VALIDATE` option for restoration causes RMAN to perform all the steps of a restore, up to the point where it is about to write the restored data to disk; then the data is discarded, and nothing is written. This is often used to validate the integrity of backup media, but is also useful to diagnose performance problems with the read and processing operations of a `RESTORE` command. `RESTORE VALIDATE` performs both decryption and decompression if the backup was created with those options.

Diagnosing Performance Bottlenecks

1. Query the `EFFECTIVE_BYTES_PER_SECOND` column in `V$BACKUP_ASYNC_IO` or `V$BACKUP_SYNC_IO` for the AGGREGATE row.
2. If the value in `EFFECTIVE_BYTES_PER_SECOND` < storage media throughput, execute the `BACKUP VALIDATE` command to obtain additional information.



Copyright © 2020, Oracle and/or its affiliates.

1. You can use the `BACKUP VALIDATE` command to help you determine whether your bottleneck is in the read or write phase. Start by querying the `EFFECTIVE_BYTES_PER_SECOND` column of `V$BACKUP_ASYNC_IO` or `V$BACKUP_SYNC_IO`.
2. If the value in `EFFECTIVE_BYTES_PER_SECOND` is less than the expected throughput from your storage media, execute the `BACKUP VALIDATE` command. `BACKUP VALIDATE` performs the same disk reads as a backup, but does not perform I/O to an output device. So by comparing the time of your backup operations with the time taken by the `BACKUP VALIDATE` command, you should be able to determine whether the bottleneck is due to reads or writes.

Diagnosing Performance Bottlenecks: Read Phase

- If BACKUP VALIDATE time \approx actual backup time, the read phase is the likely bottleneck.
- Implement appropriate RMAN multiplexing and buffer usage guidelines.
- Investigate “slow” performing files: Find the data file with the highest LONG_WAITS/IO_COUNT ratio.
 - If ASM, add disk spindles and/or rebalance disks.
 - Move the file to a new disk or multiplex with another “slow” file.



Copyright © 2020, Oracle and/or its affiliates.

If the execution time of the BACKUP VALIDATE command approximates the actual backup time, the read phase is most likely the bottleneck.

You can improve backup performance by adjusting the multiplexing level.

Is There a “Write” Problem?

To analyze a write process to disk:

- Create a data file on the disk and time the operation
- Invoke the write by calling the `DBMS_BACKUP_RESTORE.SETPARMS` function



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

To analyze a write process to disk (for both backup and restore), create a new tablespace with a data file on the disk and time the operation. If the same issues occur, general Oracle performance to this device should be tuned, not RMAN-specific parameters.

If this technique is not suitable, a write I/O driver can be used. The write driver is invoked by calling the `DBMS_BACKUP_RESTORE.SETPARMS` function with parameters:

- `p0 => 6`
- `p1 => buffer size in bytes (specify NULL or 0 to use default)`
- `p2 => number of buffers (specify NULL or 0 to use default)`
- `p3 => number of blocks to write`
- `p4 => block size in bytes (must be specified, 8192 is a good choice)`
- `p5 => file name to write`
- `p6 => 1`

The write I/O driver is also helpful when you are tuning disk channel output by executing commands with varying buffer sizes and counts.

Diagnosing Performance Bottlenecks: Write or Copy Phase

- If BACKUP VALIDATE time is less than the actual backup time, buffer copy or write to storage is the likely bottleneck.
- Implement backup compression and encryption guidelines:
 - Verify that uncompressed backup performance scales properly, as channels are added.
 - Use the `LOW` or `MEDIUM` setting.
 - Use the `AES128` encryption algorithm.
- If tape backup, check media management (MML) settings:
 - TCP/IP buffer size
 - Media management client/server buffer size
 - Client/socket timeout
 - Media server hardware, connectivity to tape
 - Enable tape compression (but not RMAN compression)



Copyright © 2020, Oracle and/or its affiliates.

If the execution time for the `BACKUP VALIDATE` command is significantly less than the actual backup time, writing to the output device is most likely the bottleneck.

If you are using compression, set the compression to `LOW` or `MEDIUM`.

Use the `AES128` encryption algorithm because it is the least CPU intensive.

Using Dynamic Views to Diagnose RMAN Performance

Use the following views to determine where RMAN backup and restore operations are encountering performance issues:

View	Use
<code>V\$SESSION_LONGOPS</code>	Monitoring the progress of backups and restore jobs
<code>V\$BACKUP_SYNC_IO</code>	Identifying bottlenecks Determining whether the tape is streaming when the I/O is synchronous Viewing detailed progress of backup jobs
<code>V\$BACKUP_ASYNC_IO</code>	Identifying bottlenecks Determining the rate of asynchronous I/O



Copyright © 2020, Oracle and/or its affiliates.

If you experience performance issues with backup and restore jobs, you may begin diagnosing your issues by using the views listed in the slide. Additional information about each view follows in the lesson.

Monitoring RMAN Job Progress

Monitor the progress of backup and restore operations by querying V\$SESSION_LONGOPS.

```
SQL> SELECT SID, SERIAL#, CONTEXT, SOFAR, TOTALWORK,
  2  ROUND(SOFAR/TOTALWORK*100, 2) "%_COMPLETE"
  3  FROM V$SESSION_LONGOPS
  4  WHERE OPNAME LIKE 'RMAN%'
  5  AND OPNAME NOT LIKE '%aggregate%'
  6  AND TOTALWORK != 0
  7  AND SOFAR <> TOTALWORK;

  SID SERIAL# CONTEXT      SOFAR TOTALWORK %_COMPLETE
-----  -----  -----  -----  -----  -----
  13      75        1    9470     15360    61.65
  12      81        1   15871     28160    56.36
```



Copyright © 2020, Oracle and/or its affiliates.

Monitor the progress of backups, copies, and restores by querying the V\$SESSION_LONGOPS view. RMAN uses detail and aggregate rows in V\$SESSION_LONGOPS. Detail rows describe the files that are being processed by one job step. Aggregate rows describe the files that are processed by all job steps in an RMAN command. A job step is the creation or restoration of one backup set or data file copy. The detail rows are updated with every buffer that is read or written during the backup step, so their granularity of update is small. The aggregate rows are updated when each job step is completed, so their granularity of update is large.

Note: Set the STATISTICS_LEVEL parameter to TYPICAL (the default value) or ALL to populate the V\$SESSION_LONGOPS view.

The relevant columns in V\$SESSION_LONGOPS for RMAN include:

- OPNAME: A text description of the row. Detail rows include RMAN:datafile copy, RMAN:full datafile backup, and RMAN:full datafile restore.
- CONTEXT: For backup output rows, the value of this column is 2. For all the other rows except proxy copy (which does not update this column), the value is 1.

SOFAR:

- For image copies, the number of blocks that have been read
- For backup input rows, the number of blocks that have been read from the files that are being backed up
- For backup output rows, the number of blocks that have been written to the backup piece
- For restores, the number of blocks that have been processed to the files that are being restored in this one job step
- For proxy copies, the number of files that have been copied

TOTALWORK:

- For image copies, the total number of blocks in the file
- For backup input rows, the total number of blocks to be read from all files that are processed in this job step
- For backup output rows, the value is 0 because RMAN does not know how many blocks it will write into any backup piece
- For restores, the total number of blocks in all files restored in this job step
- For proxy copies, the total number of files to be copied in this job step

Identifying Backup and Restore Bottlenecks

- The following views can be used to determine the source of bottlenecks and to view backup job progress:
 - V\$BACKUP_SYNC_IO
 - V\$BACKUP_ASYNC_IO
- The following rows exist for a backup or restore:
 - One row for each data file
 - One aggregate data file row
 - One row for each backup piece



Copyright © 2020, Oracle and/or its affiliates.

The maximum backup speed is limited by the available hardware. It is not possible to back up any faster than the aggregate tape bandwidth. One exception to this is if there are many empty blocks in the data files that need not be backed up.

One of the components of the backup system will be a bottleneck—which one depends on the relative speeds of the disk, tape drive, and any other transport components such as the network. As an example, if the bottleneck is the tape drive, and the tape is streaming, then the backup cannot possibly proceed any faster.

Asynchronous I/O Bottlenecks

- Use V\$BACKUP_ASYNC_IO to monitor asynchronous I/O.
- The file that has the largest ratio of LONG_WAITS to IO_COUNT is probably the bottleneck.
 - IO_COUNT: Number of I/Os performed on the file
 - LONG_WAITS: Number of times the backup/restore process directed the OS to wait until I/O was complete
- Wait times should be zero to avoid bottlenecks.
 - SHORT_WAIT_TIME_TOTAL
 - LONG_WAIT_TIME_TOTAL



Copyright © 2020, Oracle and/or its affiliates.

You can use V\$BACKUP_ASYNC_IO to monitor asynchronous I/O. The LONG_WAITS column shows the number of times the backup or restore process directed the operating system to wait until an I/O was complete. The SHORT_WAITS column shows the number of times the backup/restore process made an operating system call to poll for I/O completion in nonblocking mode. On some platforms, the asynchronous I/O implementation may cause the calling process to wait for the I/O to complete while performing a nonblocking poll for I/O.

The simplest way to identify the bottleneck is to query V\$BACKUP_ASYNC_IO for the data file that has the largest ratio for LONG_WAITS divided by IO_COUNT.

Synchronous I/O Bottlenecks

- Synchronous I/O is considered to be a bottleneck.
- Query the `DISCRETE_BYTES_PER_SECOND` column from `V$BACKUP_SYNC_IO` to view the I/O rate.
 - Compare this rate with the device's maximum rate.
 - If the rate is lower than what the device specifies, this is a tuning opportunity.



Copyright © 2020, Oracle and/or its affiliates.

When using synchronous I/O, it can easily be determined how much time the backup jobs require because devices perform only one I/O task at a time. Oracle I/O uses a polling mechanism rather than an interrupt mechanism to determine when each I/O request completes. Because the backup or restore process is not immediately notified of I/O completion by the operating system, you cannot determine the duration of each I/O.

Use `V$BACKUP_SYNC_IO` to determine the source of backup or restore bottlenecks and to determine the progress of backup jobs. `V$BACKUP_SYNC_IO` contains rows when the I/O is synchronous to the process (or thread, on some platforms) that is performing the backup.

Tuning RMAN Backup Performance

To tune RMAN backup performance, perform the following steps:

1. Remove RATE settings from configured and allocated channels.
2. Set the DBWR_IO_SLAVES parameter if you use synchronous disk I/O.
3. Set the LARGE_POOL_SIZE initialization parameter.
4. Tune the RMAN read, write, and copy phases.

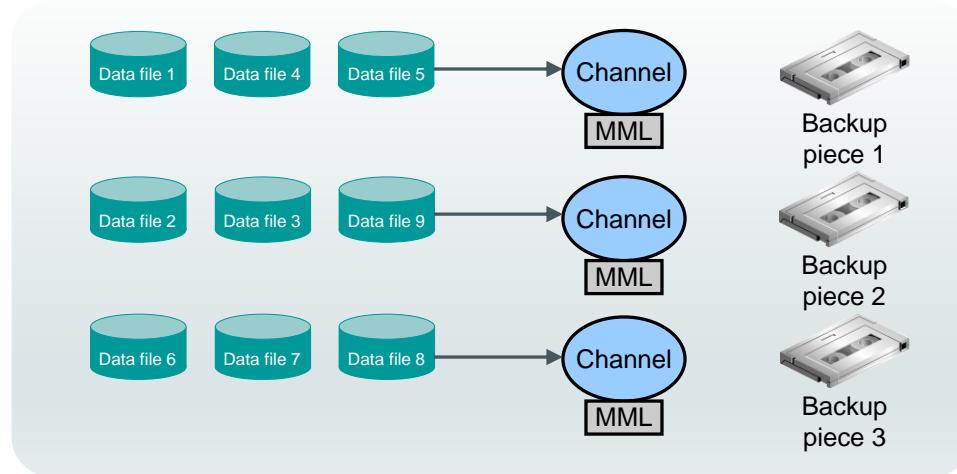


Copyright © 2020, Oracle and/or its affiliates.

1. Remove RATE settings from configured and allocated channels. The RATE parameter is used to set the maximum number of bytes (default), kilobytes (K), megabytes (M), or gigabytes (G) that RMAN reads each second on the channel. It sets an upper limit for bytes read so that RMAN does not consume too much disk bandwidth and degrade performance. If your backup is not streaming to tape, ensure that the RATE parameter is not set on the ALLOCATE CHANNEL or CONFIGURE CHANNEL command.
2. Set DBWR_IO_SLAVES if you use synchronous disk I/O. If your disk does not support asynchronous I/O, try setting the DBWR_IO_SLAVES initialization parameter to a nonzero value. Any nonzero value for DBWR_IO_SLAVES causes a fixed number (four) of disk I/O slaves to be used for backup and restore, simulating asynchronous I/O. If I/O slaves are used, I/O buffers are obtained from the SGA. The large pool is used if configured. Otherwise, the shared pool is used.
Note: By setting DBWR_IO_SLAVES, the database writer processes will use slaves as well. You may need to increase the value of the PROCESSES initialization parameter.
3. If there is failure in shared memory allocation, set the LARGE_POOL_SIZE initialization parameter as described in the topic "Setting LARGE_POOL_SIZE."
4. Tune RMAN read, write, and copy phases.

Parallelization of Backup Sets

For performance, allocate multiple channels and assign files to specific channels.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

You can configure parallel backups by setting the PARALLELISM option of the CONFIGURE command to greater than 1 or by manually allocating multiple channels. RMAN parallelizes its operation and writes multiple backup sets in parallel. The server sessions divide the work of backing up the specified files.

Example

```
RMAN> RUN {  
2>   ALLOCATE CHANNEL c1 DEVICE TYPE sbt;  
3>   ALLOCATE CHANNEL c2 DEVICE TYPE sbt;  
4>   ALLOCATE CHANNEL c3 DEVICE TYPE sbt;  
5>   BACKUP  
6>   INCREMENTAL LEVEL = 0  
7>   (DATAFILE 1,4,5 CHANNEL c1)  
8>   (DATAFILE 2,3,9 CHANNEL c2)  
9>   (DATAFILE 6,7,8 CHANNEL c3);  
10>  SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';  
11> }
```

When backing up data files, you can specify the files to be backed up by either their path name or their file number. For example, the following two commands perform the same action:

```
BACKUP DEVICE TYPE sbt DATAFILE '/home/oracle/system01.dbf';  
BACKUP DEVICE TYPE sbt DATAFILE 1;
```

When you create multiple backup sets and allocate multiple channels, RMAN automatically parallelizes its operation and writes multiple backup sets in parallel. The allocated server sessions share the work of backing up the specified data files, control files, and archived redo logs. You cannot stripe a single backup set across multiple channels.

Parallelization of backup sets is achieved by:

- Configuring PARALLELISM to greater than 1 or allocating multiple channels
- Specifying many files to back up

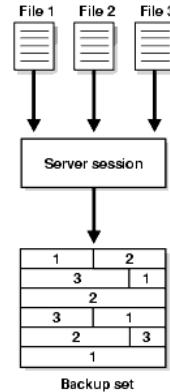
Example

- There are nine files that need to be backed up (data files 1 through 9).
- Assign the data files to a backup set so that each set has approximately the same number of data blocks to back up (for efficiency).
 - Data files 1, 4, and 5 are assigned to backup set 1.
 - Data files 2, 3, and 9 are assigned to backup set 2.
 - Data files 6, 7, and 8 are assigned to backup set 3.

Note: You can also use the FILESPERSET parameter to limit the number of data files that are included in a backup set.

RMAN Multiplexing

- Multiplexing level: Maximum number of files read by one channel, at any time, during backup
 - Min (MAXOPENFILES, FILESPERSET)
 - Default for MAXOPENFILES is 8.
 - Default for FILESPERSET default is 64.
- MAXOPENFILES determines the number and size of input buffers.
 - All buffers allocated from PGA, unless disk or tape I/O slaves, are enabled.



Copyright © 2020, Oracle and/or its affiliates.

RMAN uses two different types of buffers for I/O: disk and tape. RMAN multiplexing determines how RMAN allocates disk buffers. RMAN multiplexing is the number of files in a backup read simultaneously and then written to the same backup piece.

The degree of multiplexing depends on the FILESPERSET parameter of the BACKUP command as well as the MAXOPENFILES parameter of the CONFIGURE CHANNEL or ALLOCATE CHANNEL command.

Note: RMAN multiplexing is set at the channel level. For ASM or RAID1, set MAXOPENFILES to 1 or 2.

RMAN Multiplexing

- For reads:

Multiplexing Level	Allocation Rule
Level <= 4	1 MB buffers are allocated so that the total buffer size for all input files is 16 MB.
4 < Level <= 8	512 KB are allocated so that the total buffer size for all files is less than 16 MB.
Level > 8	RMAN allocates four 128 KB disk buffers per channel for each file so that the total size is 512 KB per channel for each file.

- For writes, each channel is allocated four output buffers of 1 MB each.



Copyright © 2020, Oracle and/or its affiliates.

For example, assume that you back up two data files with one channel. You set FILESPERSET to 3 and MAXOPENFILES to 8. In this case, the number of files in each backup set is 2 (the lesser of FILESPERSET and the files read by each channel) and the level of multiplexing is 2 (the lesser of MAXOPENFILES and the number of files in each backup set). When RMAN backs up from disk, it uses the algorithm that is described in the table shown in the slide.

For writing, each channel allocates four output buffers of size 1 MB each.

These buffers are allocated from the PGA unless DBWR_IO_SLAVES is set to a nonzero value.

Note: For best recovery performance, do not set FILESPERSET to a value greater than 8.

Summary

In this lesson, you should have learned how to:

- Interpret the RMAN message output
- Apply best-practice tuning principles
- Diagnose RMAN performance issues



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Monitoring an RMAN Backup Job



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Recovery Catalog Overview



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to describe the use of the RMAN recovery catalog.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

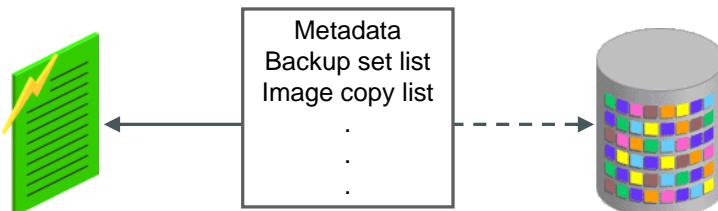
RMAN Repository Data Storage: Comparison of Options

Control file:

- Simpler administration
- Default

Recovery catalog:

- Replicates control file data
- Stores more backup history
- Services many targets
- Stores RMAN scripts
- Provides more protection options for metadata



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

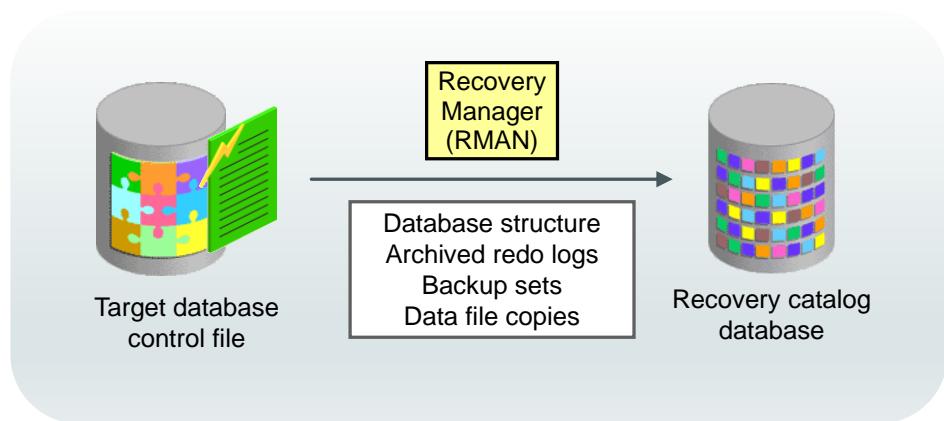
The RMAN repository data is always stored in the control file of the target database. But it can also, additionally, be stored in a separate database in a recovery catalog.

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

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

A recovery catalog is required when you use RMAN in a Data Guard configuration.

Storing Information in the Recovery Catalog



Copyright © 2020, Oracle and/or its affiliates.

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

Note: The RMAN client must be connected to the target database instance and to the catalog database instance for information propagation to take place.

Reasons to Use a Recovery Catalog

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



Copyright © 2020, Oracle and/or its affiliates.

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

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

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

Note: Enterprise Manager Cloud Control also enables you to view backup information for multiple databases without the use of a recovery catalog.

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

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

Summary

In this lesson, you should have learned how to describe the use of the RMAN recovery catalog.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Creating a Recovery Catalog

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

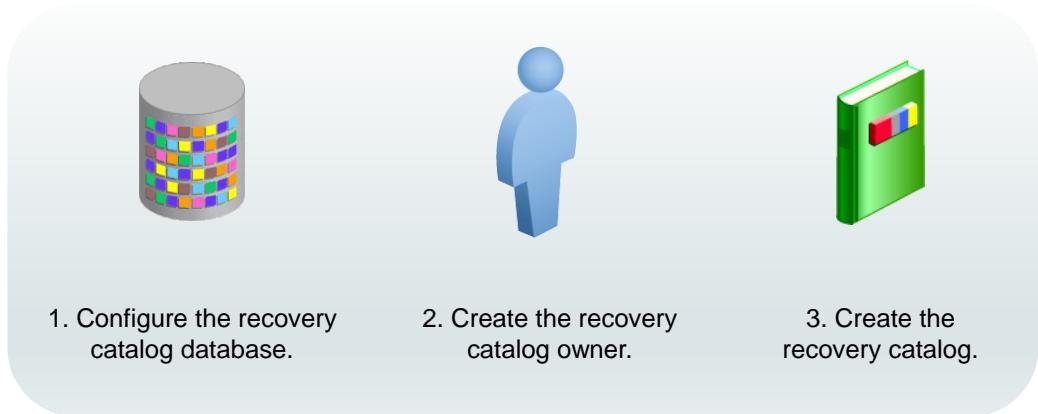
After completing this lesson, you should be able to create the RMAN recovery catalog.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Creating the Recovery Catalog: Three Steps



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

1. Configuring the Recovery Catalog Database

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

```
SQL> CREATE TABLESPACE rcat_ts DATAFILE <data file name>
2  SIZE 15M;
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Determine the database in which you will install the recovery catalog schema. Be sure to consider your backup and recovery procedures for this database.

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

2. Creating the Recovery Catalog Owner

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

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



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Create a user to serve as the recovery catalog owner. Set the default tablespace for this user to the tablespace you created for the recovery catalog. Be sure to provide UNLIMITED quota on this tablespace for the user. After you have created the user, grant the RECOVERY_CATALOG_OWNER role to the user.

The RECOVERY_CATALOG_OWNER role provides privileges for the owner of the recovery catalog. The role includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE PROCEDURE, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE TRIGGER, CREATE TYPE, and CREATE VIEW.

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

3. Creating the Recovery Catalog

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

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

- Execute the CREATE CATALOG command:

```
RMAN> CREATE CATALOG;
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

Summary

In this lesson, you should have learned how to create the RMAN recovery catalog.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Creating a Recovery Catalog Owner
- Creating the Recovery Catalog



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Managing Target Database Records

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Register a database in the RMAN recovery catalog
- Manually resynchronize the recovery catalog



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Managing Target Database Records in the Recovery Catalog

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



Copyright © 2020, Oracle and/or its affiliates.

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

Registering a Database in the Recovery Catalog

- RMAN does the following when a database is registered:
 - Creates rows in the recovery catalog tables for the target database
 - Copies data from the target database control file to the recovery catalog tables
 - Synchronizes the recovery catalog with the control file
- Using the RMAN command line to register a database:

```
$ rman TARGET / CATALOG  
      username/password@net_service_name  
RMAN> REGISTER DATABASE;
```
- Using Enterprise Manager to register a database:
 1. Navigate to the Recovery Catalog Settings page.
 2. Add the recovery catalog to the configuration if not present.
 3. Specify the target database to use the recovery catalog.



Copyright © 2020, Oracle and/or its affiliates.

After creating the recovery catalog, you must register the target databases in the recovery catalog.

To register your target database by using the RMAN command line, perform the following steps:

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

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

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

```
RMAN> REGISTER DATABASE;
```

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

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

Unregistering a Target Database from the Recovery Catalog

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

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



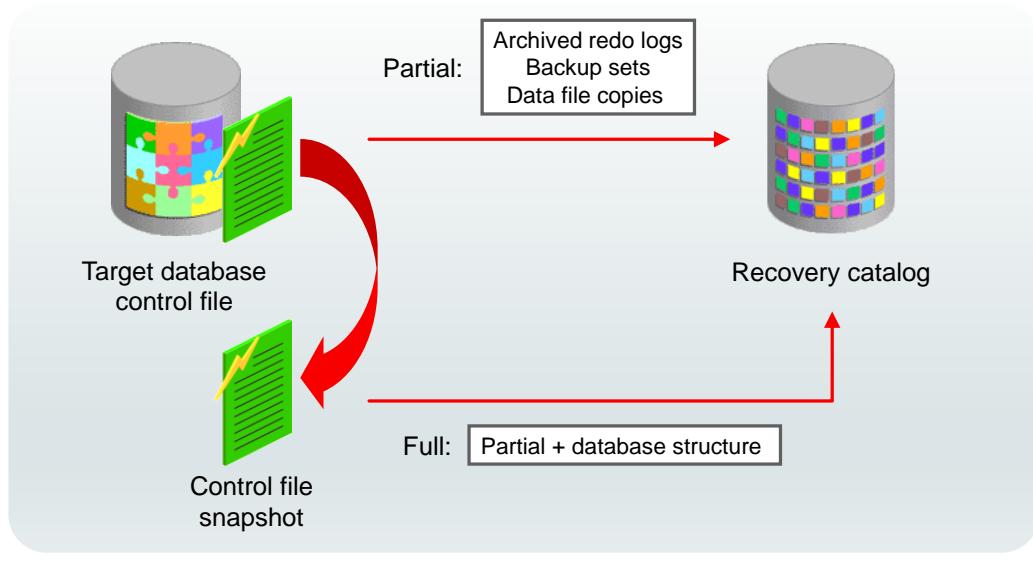
Copyright © 2020, Oracle and/or its affiliates.

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

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

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

Recovery Catalog Resynchronization: Concepts



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

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

There are two types of resynchronization: partial and full.

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

You can specify the location for the snapshot control file by using the `SNAPSHOT CONTROLFILE NAME` configuration setting. The default value for the snapshot control file name is platform-specific and depends on the Oracle home of each target database.

In an Oracle RAC configuration, the snapshot control file needs to be globally available to all instances in the RAC configuration. Refer to *Oracle Real Application Clusters Administration and Deployment Guide* for additional information on configuring the snapshot control file location in a RAC configuration.

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

The `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter determines the minimum number of days that records are retained in the control file before they are candidates for being overwritten. Ensure that `CONTROL_FILE_RECORD_KEEP_TIME` is set to a value that is longer than the interval between resynchronizations. If the value of `CONTROL_FILE_RECORD_KEEP_TIME` is less than this interval, control file records could be reused before they are propagated to the recovery catalog.

Manually Resynchronizing the Recovery Catalog

Manually resynchronize the recovery catalog in the following situations:

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

```
RMAN> RESYNC CATALOG;
```



Copyright © 2020, Oracle and/or its affiliates.

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

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

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

Summary

In this lesson, you should have learned how to:

- Register a database in the RMAN recovery catalog
- Manually resynchronize the recovery catalog



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Registering a Database in the Recovery Catalog
- Configuring the Recovery Catalog for Recovery



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Using Stored Scripts

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Describe the use of the RMAN stored scripts
- Execute RMAN stored scripts
- Maintain RMAN stored scripts



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Using RMAN Stored Scripts

Stored scripts are:

- An alternative to command files
- Available to any RMAN client that can connect to the target database and recovery catalog
- Of two types:
 - Local: Associated with the target database to which RMAN is connected when the script is created
`CREATE SCRIPT script_name { <RMAN commands>}`
 - Global: Can be executed against any database registered in the recovery catalog
`CREATE GLOBAL SCRIPT script_name { <RMAN commands>}`
- Created from a text file (additional option)
`CREATE [GLOBAL] SCRIPT script_name FROM FILE 'file_name'`



Copyright © 2020, Oracle and/or its affiliates.

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

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

Creating RMAN-Stored Scripts

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

Executing RMAN Stored Scripts

- Executing a script:

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

- Executing a global script:

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



Copyright © 2020, Oracle and/or its affiliates.

Connect to the target database and recovery catalog and use the EXECUTE SCRIPT command to execute a stored script. Note that the EXECUTE SCRIPT command requires a RUN block. If an RMAN command in the script fails, subsequent RMAN commands in the script do not execute.

When you execute the script, it uses the automatic channels configured at the time. Use ALLOCATE CHANNEL commands in the script if you need to override the configured channels as shown in the following example:

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

Maintaining RMAN Stored Scripts

- Displaying a script:

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

- Sending the contents of a script to a file:

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

- Displaying the names of defined scripts:

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

- Updating a script:

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

- Updating a script from a file:

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

- Deleting a script:

```
DELETE SCRIPT script_name;
```

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

Summary

In this lesson, you should have learned how to:

- Describe the use of the RMAN stored scripts
- Execute RMAN stored scripts
- Maintain RMAN stored scripts



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Creating a Stored Script
- Executing a Stored Script



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Creating and Using Virtual Private Catalogs

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

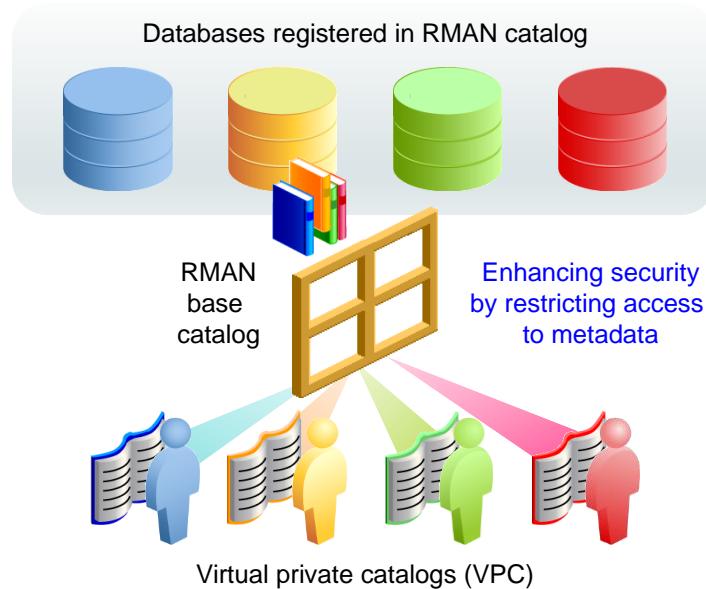
- Create a virtual private catalog
- Manage a virtual private catalog



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Creating and Using Virtual Private Catalogs



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

The RMAN catalog includes functionality to create virtual private RMAN catalogs (VPC) for groups of databases and users. The RMAN recovery catalog is created and managed by using Oracle Virtual Private Database (VPD), providing better performance and scalability when a large number of virtual private catalogs are created.

The catalog owner can grant access to a registered database and grant the REGISTER privilege to the virtual catalog owner. The virtual catalog owner can then connect to the catalog for a particular target or register a target database. After this configuration, the VPC owner uses the virtual private catalog just like a standard base catalog.

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

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

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

Creating a Virtual Private Catalog

1. Create the recovery catalog, tablespace, and catalog owner, as shown previously.
2. Create the owner of the Virtual Private Catalog (VPC).
3. Grant create session to VPC owner.
4. In RMAN, as the catalog owner, create the catalog.
5. In SQL*Plus, as sysdba, execute dbmsrmanvpc.sql to enable VPD.
6. In RMAN, as the catalog owner, connect to the catalog and upgrade the catalog.
7. Grant privileges (access to the metadata/ability to register new target databases) to the VPC owner.



Copyright © 2020, Oracle and/or its affiliates.

Virtual private catalogs are implemented through Oracle Virtual Private Database (VPD). Oracle Virtual Private Database (VPD) provides an ability to create security policies to control database access at the row and column level. Refer to *Oracle Database Security Guide* for detailed information about Oracle VPD.

These steps work for non-CDB and PDB installations, the only difference is in the connection strings. For a PDB installation, connect to the PDB instead of the CDB to perform the steps shown here.

Refer to *Oracle Database Backup and Recovery User's Guide* for additional information, including syntax examples.

Creating a Virtual Private Catalog

8. Register a database with a VPC and store backup metadata.
 - A. Using RMAN, connect to the recovery catalog database as the VPC owner and connect to the database that you want to register as TARGET.
 - B. Register the database with the VPC owner by using the REGISTER DATABASE command.
 - C. Use the BACKUP command to back up the database and store metadata related to the backup in the VPC.



Copyright © 2020, Oracle and/or its affiliates.

Managing Virtual Private Catalogs

- If needed, revoke access to the metadata:

```
RMAN> REVOKE CATALOG FOR DATABASE prod1 FROM vpc1;
```

- To drop a virtual private catalog:

```
RMAN> DROP CATALOG;
```



Copyright © 2020, Oracle and/or its affiliates.

To revoke access for a specific database, connect to the recovery catalog database as the recovery catalog owner:

```
RMAN> REVOKE CATALOG FOR DATABASE prod1 FROM vpc1;
```

To drop a virtual private catalog, connect to the recovery catalog database as the virtual private catalog owner:

```
RMAN> DROP CATALOG;
```

Upgrading Virtual Private Catalogs

1. Using SQL*Plus, connect to the recovery catalog database as the `SYS` user with the `SYSDBA` privilege.
2. Grant additional privileges to the `RECOVERY_CATALOG_OWNER` role by executing the `$ORACLE_HOME/rdbms/admin/dbmsrmansys.sql` script.
3. Using RMAN, connect to the base recovery catalog and upgrade the catalog by using the `UPGRADE CATALOG` command.
4. Using SQL*Plus, again connect to the recovery catalog database as the `SYS` user with the `SYSDBA` privilege.
5. Upgrade the VPC schemas to the VPD model by executing the `$ORACLE_HOME/rdbms/admin/dbmsrmanvpc.sql` script.



Copyright © 2020, Oracle and/or its affiliates.

RMAN uses Oracle Virtual Private Database (VPD) to implement virtual private catalogs beginning with Oracle Database 12c Release 1 (12.1.0.2). If you have a recovery catalog and virtual private catalogs using an earlier release, you must upgrade to VPD as described in the slide. Sample syntax can be found in *Oracle Database Backup and Recovery User's Guide*.

Summary

In this lesson, you should have learned how to:

- Create a virtual private catalog
- Manage a virtual private catalog



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Enabling the Virtual Private Database (VPD) Functionality
- Creating a Virtual Private Catalog
- Backing Up a PDB



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Restore and Recovery Concepts

The Oracle logo, consisting of the word "ORACLE" in white capital letters inside a red square.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Explain how to employ the best Oracle Database recovery technology for your failure situation
- Describe instance or crash recovery
- Describe complete recovery
- Describe point-in-time recovery
- Describe recovery with RESETLOGS



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

File Loss

- File loss can be caused by:
 - User error
 - Application error
 - Media failure
- A *noncritical file* loss is one where the database remains open and available.
- The loss of a noncritical file can be addressed by:
 - Creating a new file
 - Rebuilding the file
 - Recovering the lost or damaged file



Copyright © 2020, Oracle and/or its affiliates.

Files can be lost or damaged due to:

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

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

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

- The loss of an index tablespace can cause applications and queries to run much slower, or even make the application unusable, if the indexes were used to enforce constraints.
- The loss of an online redo log group, as long as it is not the current online log group, can cause database operations to be suspended until new log files are generated.

Data Repair Techniques

To respond to potential data loss:

- Physical failure (missing or corrupted data file):
 - Data Recovery Advisor
 - Data File Media Recovery
 - Block Recovery
- Logical failure (application or user error):
 - Logical Flashback Features
 - Oracle Flashback Database
 - Point-in-Time Recovery:
 - Database Point-in-Time Recovery (DBPITR)
 - Tablespace Point-in-Time Recovery (TSPITR)
 - Table Point-in-Time Recovery (TPITR)



Copyright © 2020, Oracle and/or its affiliates.

The Data Recovery Advisor can diagnose failures, advise you on how to respond to them, and repair the failures automatically.

Data file media recovery is a form of media recovery that enables you to restore data file backups and apply archived redo logs or incremental backups to recover lost changes. You can recover either a whole database or a subset of the database. Data file media recovery is the most general-purpose form of recovery and can protect against both physical and logical failures.

Block media recovery is a form of media recovery that enables you to recover individual blocks within a data file rather than the whole data file.

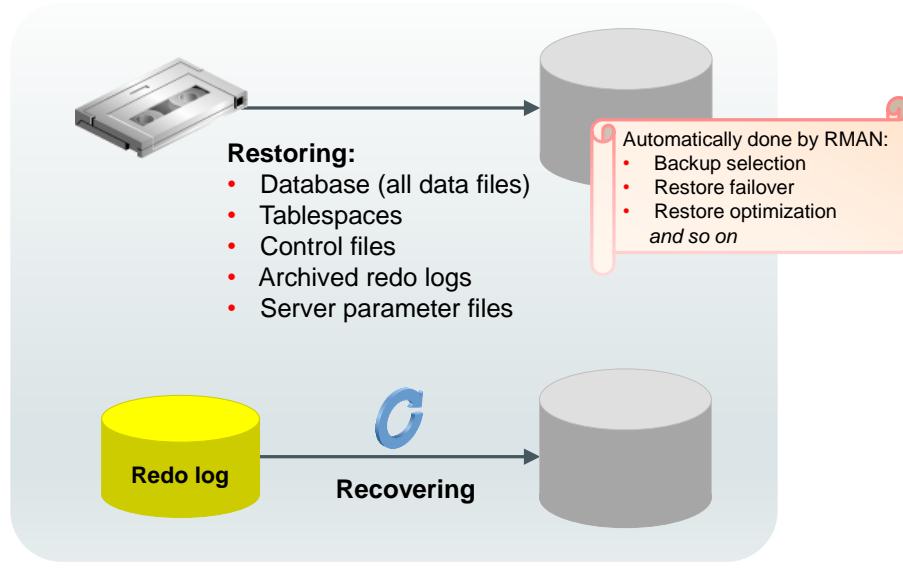
Logical flashback features enable you to view or rewind individual database objects or transactions to a past time. These features do not require the use of RMAN.

Oracle Flashback Database is a block-level recovery mechanism that is similar to media recovery, but is generally faster and does not require a backup to be restored. You can return your whole database to a previous state without restoring old copies of your data files from backup, if you have enabled flashback logging in advance. You must have a fast recovery area configured for logging for flashback database or guaranteed restore points.

Point-in-time recovery (PITR) is a specialized form of recovery in which you recover to a noncurrent time, also known as incomplete recovery.

- RMAN DBPITR restores the database from backups before the target time for recovery and then uses incremental backups and redo to roll the database forward to the target time.
- RMAN TSPITR recovers a tablespace to a time earlier than the rest of the database.
- RMAN TPITR recovers a table to an earlier point in time. Both TSPITR and TPITR use auxiliary database for the execution of these tasks.

Restoring and Recovering



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Restoring a file is the process of copying a backup into place to be used by the database. This is necessary if, for example, a file is damaged because the physical disk it is on fails. This is usually due to hardware problems, such as disk write errors or controller failure. In that case, a backup of the file needs to be copied on to a new (or repaired) disk. The file types that can be restored are listed in the slide.

- RMAN uses the records of available backup sets or image copies in the RMAN repository to select the best available backups. If two backups are from the same point in time, then RMAN prefers image copies over backup sets because RMAN can restore them more quickly (similar for disk versus tape).
- RMAN automatically uses restore failover to skip corrupted or inaccessible backups and look for usable backups.
- By default, RMAN skips restoring a data file if the file is present in the correct location and its header contains the expected information, and so on.

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

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

Using RMAN RESTORE and RECOVER Commands

- **RESTORE command:** Restores database files from backup
- **RECOVER command:** Recovers the restored files by applying changes recorded in incremental backups and redo log files

```
RMAN> ALTER TABLESPACE inv_tbs OFFLINE IMMEDIATE;
RMAN> RESTORE TABLESPACE inv_tbs;
RMAN> RECOVER TABLESPACE inv_tbs;
RMAN> ALTER TABLESPACE inv_tbs ONLINE;
```

- The Enterprise Manager Cloud Control Recovery Wizard creates and runs an RMAN script to perform the recovery.



Copyright © 2020, Oracle and/or its affiliates.

Reconstructing the contents of an entire database or a part of it from a backup typically involves two phases: retrieving a copy of the data file from a backup and reapplying changes to the file since the backup from the archived and online redo logs, to bring the database to the desired SCN (usually the most recent one).

- RESTORE {DATABASE | TABLESPACE name [,name]... | DATAFILE name [,name] }...

The RESTORE command retrieves the data file on to disk from a backup location on tape, disk, or other media and makes it available to the database server. RMAN restores from backup any archived redo logs required during the recovery operation. If backups are stored on a media manager, channels must be configured or allocated for use in accessing backups stored there.

- RECOVER {DATABASE | TABLESPACE name [,name]... | DATAFILE name [,name] }...

The RECOVER command takes the restored copy of the data file and applies to it the changes recorded in the incremental backups and database's redo logs.

You can also perform complete or point-in-time recovery by using the Recovery Wizard. Navigate from the Cloud Control database home page: Availability > Backup & Recovery > Perform Recovery.

Note: An automated method of detecting the need for recovery and carrying out that recovery makes use of the Data Recovery Advisor.

Instance Failure

Typical Causes	Possible Solutions
Power outage	Restart the instance by using the STARTUP command. Recovering from instance failure is automatic, including rolling forward changes in the redo logs and then rolling back any uncommitted transactions.
Hardware failure	
Failure of one of the critical background processes	Investigate the causes of failure by using the alert log, trace files, and Cloud Control.
Emergency shutdown procedures	



Copyright © 2020, Oracle and/or its affiliates.

Instance failure occurs when the database instance is shut down before synchronizing all database files. An instance failure can occur because of hardware or software failure or through the use of the emergency SHUTDOWN ABORT and STARTUP FORCE shutdown commands.

Instance Recovery

Automatic instance or crash recovery:

- Is caused by attempts to open a database whose files are not synchronized on shutdown
- Uses information stored in redo log groups to synchronize files
- Involves two distinct operations:
 - **Rolling forward:** Redo log changes (both committed and uncommitted) are applied to data files.
 - **Rolling back:** Changes that are made but not committed are returned to their original state.

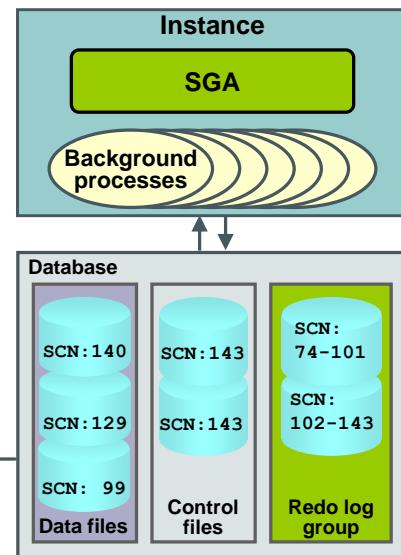
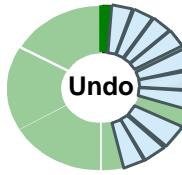


Copyright © 2020, Oracle and/or its affiliates.

The Oracle database automatically recovers from instance failure. All that needs to happen is for the instance to be started normally. The instance mounts the control files and then attempts to open the data files. When it discovers that the data files have not been synchronized during shutdown, the instance uses information contained in the redo log groups to roll the data files forward to the time of shutdown. Then the database is opened, and any uncommitted transactions are rolled back.

Phases of Instance Recovery

1. Startup instance (data files are out of sync)
2. Roll forward (redo)
3. Committed and uncommitted data in files
4. Database opened
5. Roll back (undo)
6. Committed data in files



ORACLE

Copyright © 2020, Oracle and/or its affiliates.

For an instance to open a data file, the system change number (SCN) contained in the data file's header must match the current SCN that is stored in the database's control files.

If the numbers do not match, the instance applies redo data from the online redo logs, sequentially "redoing" transactions until the data files are up to date. After all data files have been synchronized with the control files, the database is opened, and users can log in.

When redo logs are applied, *all* transactions are applied to bring the database up to the state as of the time of failure. This usually includes transactions that are in progress but have not yet been committed. After the database has been opened, those uncommitted transactions are rolled back. At the end of the rollback phase of instance recovery, the data files contain only committed data.

Media Failure

Typical Causes	Possible Solutions
Failure of disk drive	<ol style="list-style-type: none">1. Restore the affected file from backup.
Failure of disk controller	<ol style="list-style-type: none">2. Inform the database about a new file location (if necessary).
Deletion or corruption of a file needed for database operation	<ol style="list-style-type: none">3. Recover the file by applying redo information (if necessary).
Logical unit (LUN) in a storage array going offline	



Copyright © 2020, Oracle and/or its affiliates.

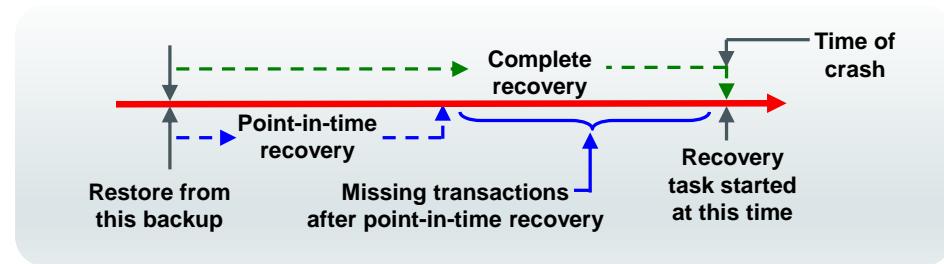
Oracle Corporation defines *media failure* as any failure that results in the loss or corruption of one or more database files (data, control, or redo log file).

Recovering from media failure requires that you restore and recover the missing files. To ensure that your database can be recovered from media failure, follow the best practices outlined in the next few slides.

Comparing Complete and Incomplete Recovery

Recovery can have two kinds of scope:

- **Complete recovery:** Brings the database or tablespace up to the point of failure, including all committed data changes made up to the point of failure
- **Incomplete or point-in-time recovery (PITR):** Brings the database or tablespace up to a specified point in time in the past, before the recovery operation was requested

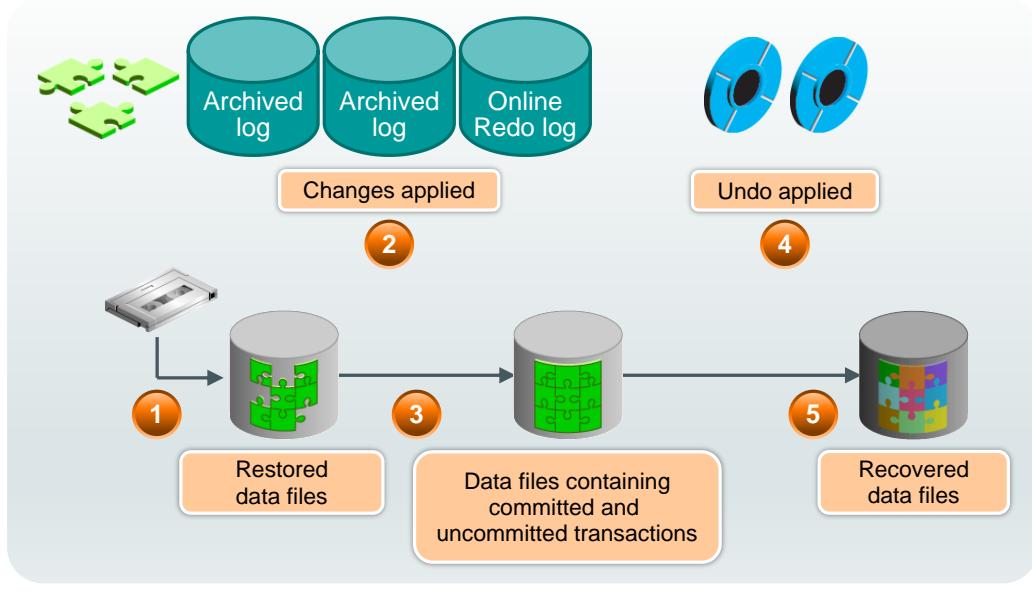


Copyright © 2020, Oracle and/or its affiliates.

When you perform a complete recovery, you bring the database to the state where it is fully up-to-date, including all committed data modifications to the point of failure.

Incomplete recovery, however, brings the database or tablespace to some point in the past point in time. This is also known as “Point-in-Time Recovery (PITR).” It means there are missing transactions; any data modifications done between the recovery destination time and the present are lost. In many cases, this is the desirable goal because there may have been some changes made to the database that need to be undone. Recovering to a point in the past is a way to remove the unwanted changes.

Complete Recovery Process



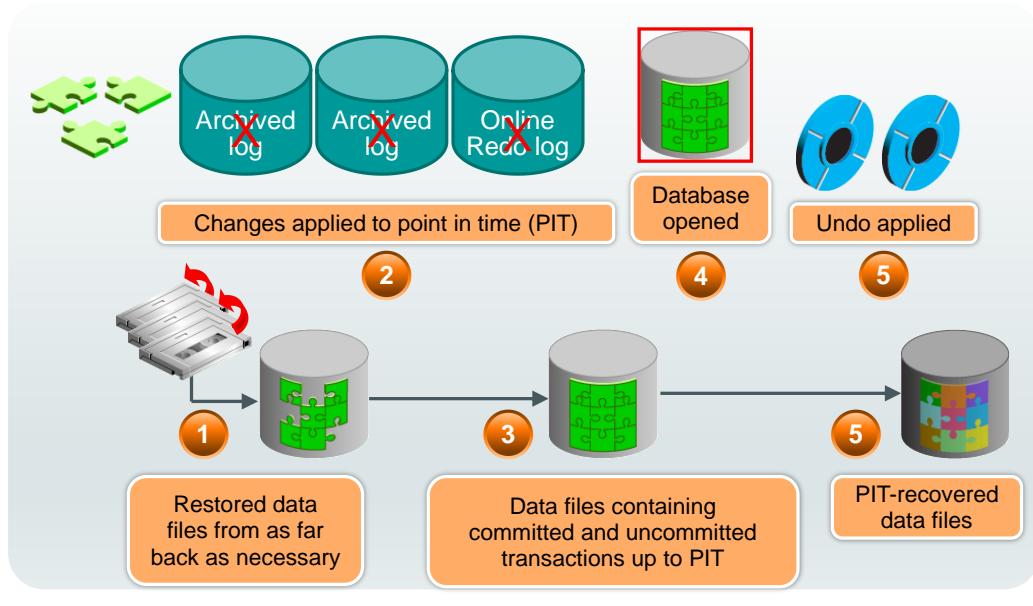
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The following steps describe what takes place during complete recovery:

1. Damaged or missing files are restored from a backup.
2. Changes from incremental backups, archived redo log files, and online redo log files are applied as necessary. The redo log changes are applied to the data files until the current online log is reached and the most recent changes have been applied. Undo blocks are generated during this entire process. This is referred to as rolling forward or cache recovery.
3. The restored data files may now contain committed and uncommitted changes.
4. The undo blocks are used to roll back any uncommitted changes. This is sometimes referred to as transaction recovery.
5. The data files are now in a recovered state and are consistent with the other data files in the database.

Point-in-Time Recovery Process



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Incomplete recovery, or database point-in-time recovery (DBPITR), uses a backup to produce a noncurrent version of the database. That is, you do not apply all of the redo records generated after the most recent backup. Perform this type of recovery only when absolutely necessary. To perform point-in-time recovery, you need:

- A valid offline or online backup of all the data files made before the recovery point
- All archived logs from the time of the backup until the specified time of recovery

The progression taken to perform a point-in-time recovery is listed as follows:

1. **Restore the data files from backup:** The backup that is used must be from before your target recovery point. This entails copying files either by using OS commands or by using the RMAN RESTORE command.
2. **Use the RECOVER command:** Apply redo from the archived redo log files, including as many as necessary to reach the restore point destination.
3. **State of over-recovery:** Now the data files contain some committed and some uncommitted transactions because the redo can contain uncommitted data.
4. **Use the ALTER DATABASE OPEN command:** The database is opened before undo is applied. This is to provide higher availability.
5. **Apply undo data:** While the redo was being applied, redo supporting the undo data files was also applied. So the undo is available to be applied to the data files in order to undo any uncommitted transactions. That is done next.
6. **Process complete:** The data files are now recovered to the point in time that you chose.

Oracle Flashback Database is the most efficient alternative to DBPITR. Unlike the other flashback features, it operates at a physical level and reverts the current data files to their contents at a past time. The result is like the result of a DBPITR, including OPEN RESETLOGS, but Flashback Database is typically faster because it does not require you to restore data files and requires only limited application of redo compared to media recovery.

For Instructor Use Only.
This document should not be distributed.

Recovery with the RESETLOGS Option

- Issue: Missing archive logs for target recovery SCN
- Workflow:
 1. Restore backups.
 2. Recover as far forward as the unbroken series of archive logs allows.
 3. Open the database with the `RESETLOGS` option.

A new **database incarnation** is automatically created to avoid confusion when two different redo streams have the same SCNs, but occurred at different times.

Note: Changes after the last applied archive log **are lost**.



Copyright © 2020, Oracle and/or its affiliates.

PITR is the only option if you must perform a recovery and discover that you are missing an archived log containing transactions that occurred sometime between the time of the backup you are restoring from and the target recovery SCN. Without the missing log, you have no record of the updates to your data files during that period. Your only choice is to recover the database from the point in time of the restored backup, as far as the unbroken series of archived logs permits, and then open the database with the `RESETLOGS` option. All changes in or after the missing redo log file are lost.

A “current” database incarnation is created whenever you open the database with the `RESETLOGS` option. The incarnation from which the current one is branched is called “parent incarnation.” An incarnation number is used to uniquely tag and identify a stream of redo.

After a complete recovery, you can resume normal operations without an `OPEN RESETLOGS`. After a DBPITR or recovery with a backup control file, however, you must open the database with the `RESETLOGS` option, thereby creating a new incarnation of the database. The database requires a new incarnation to avoid confusion when two different redo streams have the same SCNs, but occurred at different times. If you apply the wrong redo to your database, then you corrupt it.

The existence of multiple incarnations of a single database determines how RMAN treats backups that are not in the current incarnation path. Usually, the current database incarnation is the correct one to use.

Note: You can recover a pre-`RESETLOGS` backup like any other backup. To perform this type of recovery (referred to as recovery through `RESETLOGS` and recovery to an ancestor incarnation), you must have all archived logs generated after the most recent backup and at least one control file (current, backup, or created).

Restore and Recovery Performance: Best Practices

- Minimize the number of archive logs to be applied by using incremental backups.
 - **Cumulative incremental backups:** Only the most recent cumulative incremental backup must be applied. This reduces tape library requests for media backups.
 - **Differential incremental backups:** All differential incremental level 1 backups since the restored data file backup must be applied.
- Use block media recovery for isolated block corruptions.
- Keep an adequate number of archived logs on disk.
- Increase RMAN buffer memory usage.
- Tune the database for I/O, DBWR performance, and CPU utilization.



Copyright © 2020, Oracle and/or its affiliates.

The slide contains a list of best practices that will help to improve restore and recovery performance.

Summary

In this lesson, you should have learned how to:

- Determine the best Oracle Database recovery technology for your failure situation
- Describe instance or crash recovery
- Describe complete recovery
- Describe point-in-time recovery
- Describe recovery with RESETLOGS



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Diagnosing Failures

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Detect and repair database corruption
- Use the Automatic Diagnostic Repository
- Analyze instance recovery with ADRCI
- Use Data Recovery Advisor

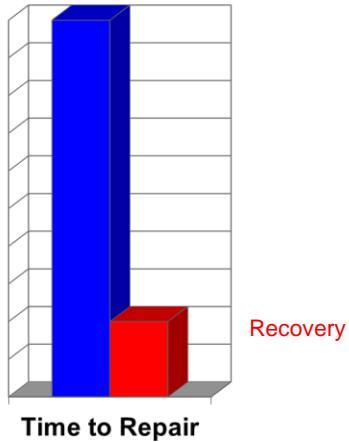


ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Reducing Problem Diagnosis Time

Investigation & Planning



Oracle tools for data repair include:

- RMAN for physical media loss or corruptions
- Flashback for logical errors
- Data Guard for physical problems
- **Data Recovery Advisor** addresses:
 - Problem diagnosis (choosing the right solution can be error prone and time-consuming)
 - Incorrect choices (errors more likely during emergencies)

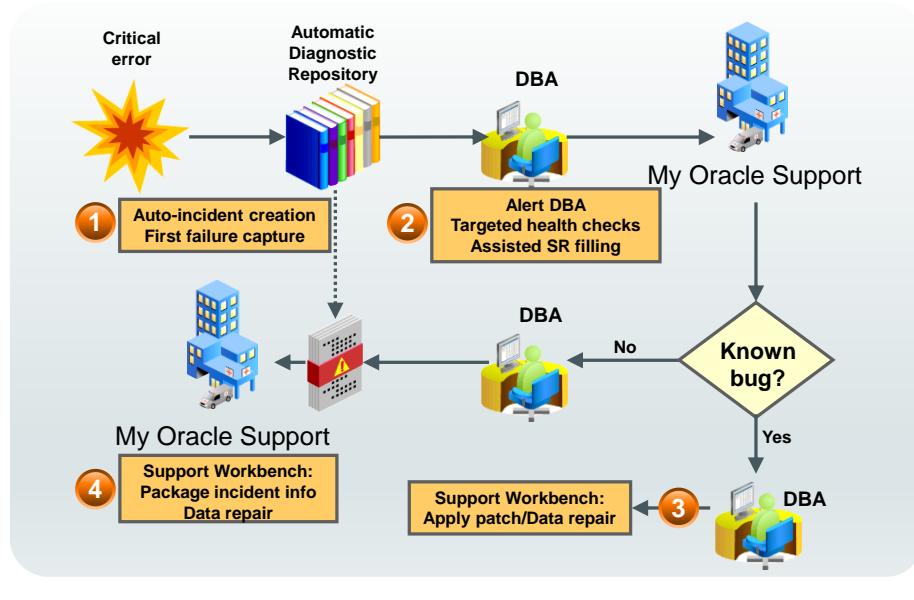


Copyright © 2020, Oracle and/or its affiliates.

Research shows that database administrators spend a large percentage of “repair time” in investigating what, why, and how data has been compromised. They may need to analyze errors, alerts, and trace files. The Data Recovery Advisor provides intelligent database problem identification and may reduce overall system downtime by eliminating or reducing the amount of time a database administrator spends researching a problem.

In addition, Oracle Data Recovery Advisor reduces uncertainty and confusion, which may often occur during an outage. The advisor uses a special repository, called Automatic Diagnostic Repository (ADR).

Automatic Diagnostic Workflow



ORACLE®

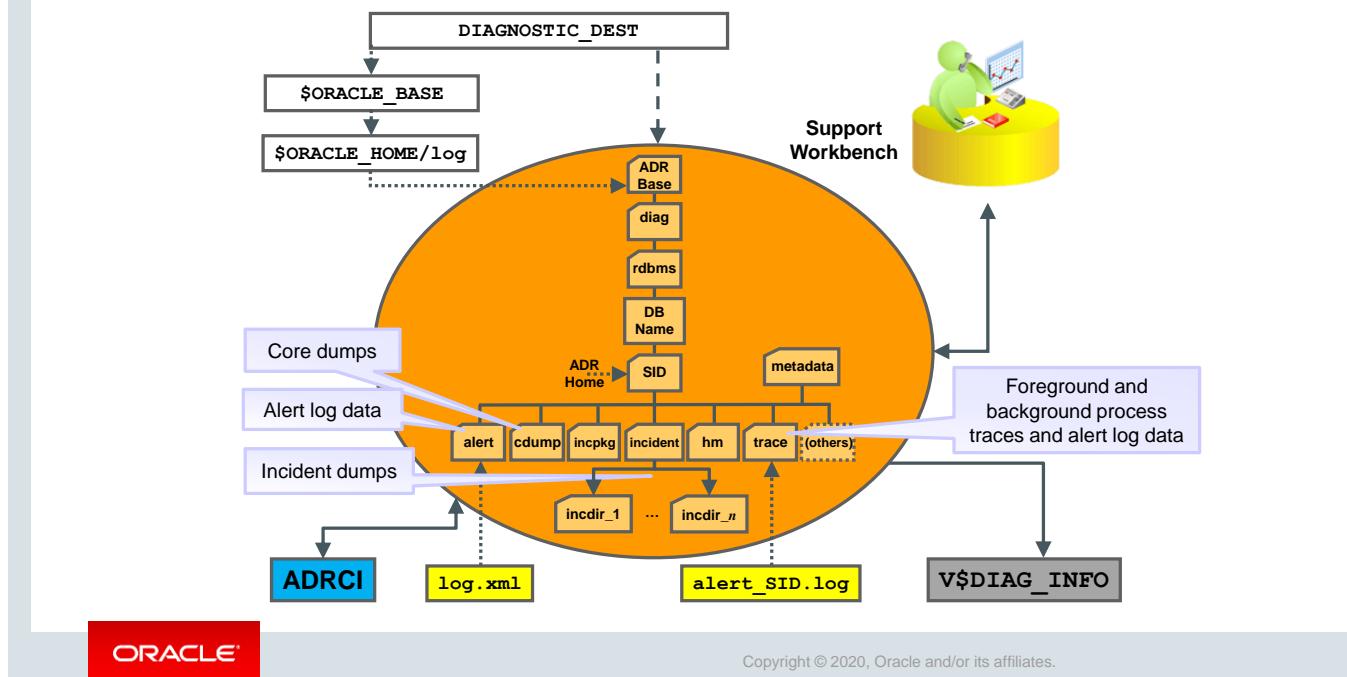
Copyright © 2020, Oracle and/or its affiliates.

An always-on, in-memory tracing facility enables database components to capture diagnostic data upon first failure for critical errors. The ADR is automatically maintained to hold diagnostic information about critical error events. This information can be used by the Data Recovery Advisor and also to create incident packages for Oracle Support Services.

Here is a typical workflow for a diagnostic session including Oracle Support Services:

1. Incident causes an alert to be raised in Enterprise Manager (EM).
2. The DBA can view the alert via the Cloud Control Alert page.
3. The DBA can drill down to incident and problem details.
4. The DBA or Oracle Support Services can decide or ask for that information to be packaged and sent to Oracle Support Services via My Oracle Support. The DBA can add files to the data to be packaged automatically.

Automatic Diagnostic Repository



ADR is a file-based repository for database diagnostic data such as traces, incident dumps and packages, the alert log, Health Monitor reports, core dumps, and more. It has a unified directory structure across multiple instances and multiple products—stored outside of any database. It is, therefore, available for problem diagnosis when the database is down.

Oracle Database server, Automatic Storage Management (ASM), Cluster Ready Services (CRS), and other Oracle products or components store all diagnostic data in the ADR. Each instance of each product stores diagnostic data underneath its own ADR home directory. For example, in a Real Application Clusters environment with shared storage and ASM, each database instance and each ASM instance have a home directory within the ADR. ADR's unified directory structure, consistent diagnostic data formats across products and instances, and a unified set of tools enable customers and Oracle Support to correlate and analyze diagnostic data across multiple instances.

The ADR root directory is known as the ADR base. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter. If this parameter is omitted or left null, the database sets `DIAGNOSTIC_DEST` upon startup as follows: If the environment variable `ORACLE_BASE` is set, `DIAGNOSTIC_DEST` is set to `$ORACLE_BASE`. If the environment variable `ORACLE_BASE` is not set, `DIAGNOSTIC_DEST` is set to `$ORACLE_HOME/log`.

ADR Command-Line Tool (ADRCI)

- ADRCI provides interaction with ADR from an operating system prompt.
- Using ADRCI, you can view diagnostic data within the ADR.

```
$ adrci
ADRCI: Release 12.1.0.1.0 - Production on Thu Nov 29 21:15:27 2012
Copyright (c) 1982, 2012, Oracle and/or its affiliates.
ADR base = "/u01/app/oracle"
ADRCI>
ADRCI> show incident
ADRCI> set editor gedit
ADRCI> show alert
.
.
ADR Home = /u01/app/oracle/diag/rdbms/em12rep/em12rep:
*****
INCIDENT_ID PROBLEM_KEY CREATE_TIME
-----
4985      ORA 4031    2012-11-21 00:57:43.823000 +00:00
5161      ORA 4031    2012-11-21 00:58:17.284000 +00:00
2 incident info records fetched
```



Copyright © 2020, Oracle and/or its affiliates.

ADRCI is a command-line tool that is part of the database fault diagnosability infrastructure. ADRCI enables you to:

- View diagnostic data within the ADR
- Package incident and problem information into a ZIP file for transmission to Oracle Support

ADRCI can be used in interactive mode or within scripts. In addition, ADRCI can execute scripts of ADRCI commands in the same way that SQL*Plus executes scripts of SQL and PL/SQL commands. There is no need to log in to ADRCI, because the data in the ADR is not intended to be secure. ADR data is secured only by operating system permissions on the ADR directories.

The easiest way to package and otherwise manage diagnostic data is with the Support Workbench of Enterprise Manager (which assists with the resolution of database errors as well as ASM errors).

ADRCI provides a command-line alternative to most of the functionality of Support Workbench and adds capabilities such as listing and querying trace files. The example in the slide shows you an ADRCI session where you are listing all open incidents stored in ADR.

Note: For more information about ADRCI and Support Workbench, refer to the *Oracle Database Utilities* guide.

V\$DIAG_INFO View

SQL> SELECT NAME, VALUE FROM V\$DIAG_INFO;	
NAME	VALUE
Diag Enabled	TRUE
ADR Base	/u01/app/oracle
ADR Home	/u01/app/oracle/diag/rdbms/orcl/orcl
Diag Trace	/u01/app/oracle/diag/rdbms/orcl/orcl/trace
Diag Alert	/u01/app/oracle/diag/rdbms/orcl/orcl/alert
Diag Incident	/u01/app/oracle/diag/rdbms/orcl/orcl/incident
Diag Cdmp	/u01/app/oracle/diag/rdbms/orcl/orcl/cdump
Health Monitor	/u01/app/oracle/diag/rdbms/orcl/orcl/hm
Default Trace File	/u01/app/oracle/diag/.../trace/orcl_ora_11424.trc
Active Problem Count	3
Active Incident Count	8



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The V\$DIAG_INFO view lists all important ADR locations:

- **ADR Base:** Path of the ADR base
 - **ADR Home:** Path of the ADR home for the current database instance
- Note:** This is a path name. There is no official environment variable called *ADR_HOME*.
- **Diag Trace:** Location of the text alert log and background/foreground process trace files
 - **Diag Alert:** Location of an XML version of the alert log
 - **Diag Incident:** Incident logs are written here.
 - **Diag Cdmp:** Diagnostic core files are written to this directory.
 - **Health Monitor:** Location of logs from Health Monitor runs
 - **Default Trace File:** Path to the trace file for your session. SQL trace files are written here.

Data Recovery Advisor

- Fast detection, analysis, and repair of failures
- Minimizing disruptions for users
- Down time and runtime failures
- User interfaces:
 - EM GUI interface (several paths)
 - RMAN command line
- Supported database configurations:
 - Single-instance
 - Not RAC
 - Supporting failover to standby, but not analysis and repair of standby databases



Copyright © 2020, Oracle and/or its affiliates.

The Data Recovery Advisor automatically gathers data failure information when an error is encountered. In addition, it can proactively check for failures. In this mode, it can potentially detect and analyze data failures before a database process discovers the corruption and signals an error. (Note that repairs are always under human control.)

Data failures can be very serious. For example, if your current log files are missing, you cannot start your database. Some data failures (such as block corruptions in data files) are not catastrophic, in that they do not take the database down or prevent you from starting the Oracle instance. The Data Recovery Advisor handles both cases: the one where you cannot start up the database (because some required database files are missing, inconsistent, or corrupted) and the one where file corruptions are discovered during run time.

User Interfaces

Data Recovery Advisor is available from Enterprise Manager (EM) Cloud Control and in RMAN. When failures exist, there are several ways to access the Data Recovery Advisor. The following examples all begin on the Database Instance home page:

- Availability > Backup & Recovery > Perform Recovery > Advise and Recover
- Active Incidents link > on the Support Workbench “Problems” page: Checker Findings tabbed page > Launch Recovery Advisor
- Database Instance Health > click the specific link, for example, ORA 1578 in the Incidents section > Support Workbench, Problems Detail page > Data Recovery Advisor
- Database Instance Health > Related Links section: Support Workbench > Checker Findings tabbed page: Launch Recovery Advisor
- Related Link: Advisor Central > Advisors tabbed page: Data Recovery Advisor
- Related Link: Advisor Central > Checkers tabbed page: Details > Run Detail tabbed page: Launch Recovery Advisor

You can also use it via the RMAN command-line, for example:

```
rman target / nocatalog  
rman> list failure all;
```

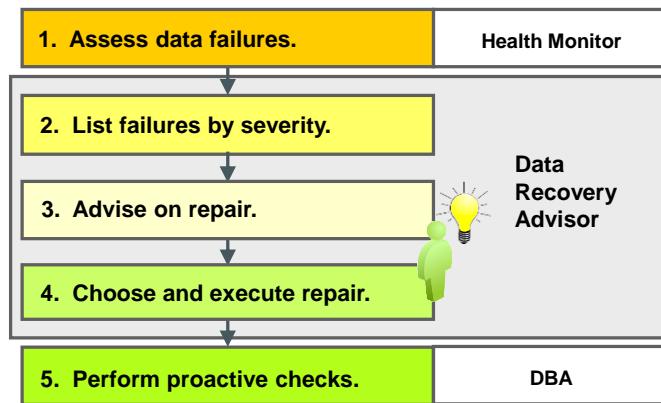
Supported Database Configurations

In the current release, the Data Recovery Advisor supports single-instance databases. Oracle Real Application Clusters (RAC) databases are not supported.

The Data Recovery Advisor cannot use blocks or files transferred from a standby database to repair failures on a primary database. Also, you cannot use the Data Recovery Advisor to diagnose and repair failures on a standby database. However, the Data Recovery Advisor does support failover to a standby database as a repair option (as mentioned earlier).

Data Recovery Advisor

Reducing down time by eliminating confusion:



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The automatic diagnostic workflow in Oracle Database performs as follows. With the Data Recovery Advisor, you need to initiate only an advice and a repair.

1. The Health Monitor automatically executes checks and logs failures and their symptoms as "findings" into the ADR.
2. The Data Recovery Advisor consolidates findings into failures. It lists the results of previously executed assessments with failure severity (critical or high).
3. When you ask for repair advice on a failure, the Data Recovery Advisor maps failures to automatic and manual repair options, checks basic feasibility, and presents you with the repair advice.
4. You can choose to manually execute a repair or request the Data Recovery Advisor to do it for you.
5. In addition to the automatic, primarily "reactive" checks of the Health Monitor and Data Recovery Advisor, Oracle recommends to additionally use the `VALIDATE` command as a "proactive" check.

Data Failure: Examples

- Not accessible components, for example:
 - Missing data files at the OS level
 - Incorrect access permissions
 - Offline tablespace and so on
- Physical corruptions, such as block checksum failures or invalid block header field values
- Logical corruptions, such as inconsistent dictionary, corrupt row piece, corrupt index entry, or corrupt transaction
- Inconsistencies, such as control file is older or newer than the data files and online redo logs
- I/O failures, such as a limit on the number of open files exceeded, channels inaccessible, network or I/O error



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Data failures are detected by checks, which are diagnostic procedures that assess the health of the database or its components. Each check can diagnose one or more failures, which are mapped to a repair.

Checks can be reactive or proactive. When an error occurs in the database, “reactive checks” are automatically executed. You can also initiate “proactive checks,” for example, by executing the `VALIDATE DATABASE` command.

In Enterprise Manager Cloud Control, navigate to the Perform Recovery page, if you find your database in a “down” or “mounted” state.

The Data Recovery Advisor can analyze failures and suggest repair options for issues, as outlined in the slide.

Data Recovery Advisor RMAN Command-Line Interface

RMAN Command	Action
LIST FAILURE	Lists previously executed failure assessment
ADVISE FAILURE	Displays recommended repair option
REPAIR FAILURE	Repairs and closes failures (after ADVISE in the same RMAN session)
CHANGE FAILURE	Changes or closes one or more failures



Copyright © 2020, Oracle and/or its affiliates.

If you suspect or know that a database failure has occurred, then use the `LIST FAILURE` command to obtain information about these failures. You can list all or a subset of failures and restrict output in various ways. Failures are uniquely identified by failure numbers. Note that these numbers are not consecutive, so gaps between failure numbers have no significance.

The `ADVISE FAILURE` command displays a recommended repair option for the specified failures. It prints a summary of the input failure and implicitly closes all open failures that are already fixed. The default behavior when no option is used is to advise on all the `CRITICAL` and `HIGH` priority failures that are recorded in ADR.

The `REPAIR FAILURE` command is used after an `ADVISE FAILURE` command **within the same RMAN session**. By default, the command uses the single, recommended repair option of the last `ADVISE FAILURE` execution in the current session. If none exists, the `REPAIR FAILURE` command initiates an implicit `ADVISE FAILURE` command. After completing the repair, the command closes the failure.

The `CHANGE FAILURE` command changes the failure priority or closes one or more failures. You can change a failure priority only for `HIGH` or `LOW` priorities. Open failures are closed implicitly when a failure is repaired. However, you can also explicitly close a failure.

Listing Data Failures

The RMAN LIST FAILURE command lists previously executed failure assessment:

- Including newly diagnosed failures
- Removing closed failures (by default)

Syntax:

```
LIST FAILURE
[ ALL | CRITICAL | HIGH | LOW | CLOSED |
  failnum[,failnum,...] ]
[ EXCLUDE FAILURE failnum[,failnum,...] ]
[ DETAIL ]
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The RMAN LIST FAILURE command lists failures. If the target instance uses a recovery catalog, it can be in STARTED mode, otherwise it must be in MOUNTED mode. The LIST FAILURE command does not initiate checks to diagnose new failures; rather, it lists the results of previously executed assessments. Repeatedly executing the LIST FAILURE command revalidates all existing failures. If the database diagnoses new ones (between command executions), they are displayed. If a user manually fixes failures, or if transient failures disappear, then the Data Recovery Advisor removes these failures from the LIST FAILURE output. The following is a description of the syntax:

- **failnum:** Number of the failure to display repair options for
- **ALL:** List failures of all priorities.
- **CRITICAL:** List failures of CRITICAL priority and OPEN status. These failures require immediate attention, because they make the whole database unavailable (for example, a missing control file).
- **HIGH:** List failures of HIGH priority and OPEN status. These failures make a database partly unavailable or unrecoverable, so they should be repaired quickly (for example, missing archived redo logs).
- **LOW:** List failures of LOW priority and OPEN status. Failures of a low priority can wait until more important failures are fixed.
- **CLOSED:** List only closed failures.

- **EXCLUDE FAILURE:** Exclude the specified list of failure numbers from the list.
- **DETAIL:** List failures by expanding the consolidated failure. For example, if there are multiple block corruptions in a file, the **DETAIL** option lists each one of them.

See the *Oracle Database Backup and Recovery Reference* for details of the command syntax.

Advising on Repair

The RMAN ADVISE FAILURE command:

- Displays a summary of input failure list
- Includes a warning, if new failures appeared in ADR
- Displays a manual checklist
- Lists a single recommended repair option
- Generates a repair script (for automatic or manual repair)

```
...
Repair script:
/u01/app/oracle/diag/rdbms/orcl/orcl/hm/reco_29791
28860.hm
RMAN>
```



Copyright © 2020, Oracle and/or its affiliates.

The RMAN ADVISE FAILURE command displays a recommended repair option for the specified failures. The ADVISE FAILURE command prints a summary of the input failure. The command implicitly closes all open failures that are already fixed.

The default behavior (when no option is used) is to advise on all the CRITICAL and HIGH priority failures that are recorded in ADR. If a new failure has been recorded in ADR since the last LIST FAILURE command, this command includes a WARNING before advising on all CRITICAL and HIGH failures.

Two general repair options are implemented: no-data-loss and data-loss repairs.

When the Data Recovery Advisor generates an automated repair option, it generates a script that shows you how RMAN plans to repair the failure. If you do not want the Data Recovery Advisor to automatically repair the failure, then you can use this script as a starting point for your manual repair. The operating system (OS) location of the script is printed at the end of the command output. You can examine this script, customize it (if needed), and also execute it manually if, for example, your audit trail requirements recommend such an action.

Syntax

```
ADVISE FAILURE
[ ALL | CRITICAL | HIGH | LOW | failnum[,failnum,...] ]
[ EXCLUDE FAILURE failnum [,failnum,...] ]
```

Executing Repairs

The RMAN REPAIR FAILURE command:

- Follows the ADVISE FAILURE command
- Repairs the specified failure
- Closes the repaired failure

Syntax:

```
REPAIR FAILURE
[USING ADVISE OPTION integer]
[ { NOPROMPT | PREVIEW} . . . ]
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

This command should be used after an ADVISE FAILURE command in the same RMAN session.

By default (with no option), the command uses the single, recommended repair option of the last ADVISE FAILURE execution in the current session. If none exists, the REPAIR FAILURE command initiates an implicit ADVISE FAILURE command.

With USING ADVISE OPTION *integer*, you specify your desired repair option by its option number (from the ADVISE FAILURE command); this is not the failure number.

By default, you are asked to confirm the command execution because you may be requesting substantial changes that take time to complete. During the execution of a repair, the output of the command indicates what phase of the repair is being executed.

After completing the repair, the command closes the failure.

You cannot run multiple concurrent repair sessions. However, concurrent REPAIR ... PREVIEW sessions are allowed.

- PREVIEW means: Do not execute the repairs; instead, display the previously generated RMAN script with all repair actions and comments.
- NOPROMPT means: Do not ask for confirmation.

Classifying (and Closing) Failures

The RMAN CHANGE FAILURE command:

- Changes the failure priority (except for CRITICAL)
- Closes one or more failures

Example:

```
RMAN> change failure 5 priority low;
List of Database Failures
=====
Failure ID Priority Status      Time Detected Summary
-----  -----  -----  -----
5        HIGH    OPEN       20-DEC-18   one or more datafiles are missing
Do you really want to change the above failures (enter YES or NO)? yes
changed 1 failures to LOW priority
```



Copyright © 2020, Oracle and/or its affiliates.

The CHANGE FAILURE command is used to change the failure priority or close one or more failures.

Syntax

```
CHANGE FAILURE
{ ALL | CRITICAL | HIGH | LOW | failnum[,failnum,...] }
[ EXCLUDE FAILURE failnum[,failnum,...] ]
{ PRIORITY {CRITICAL | HIGH | LOW} |
CLOSE } – Change status of the failure(s) to closed.
[ NOPROMPT ] – Do not ask user for a confirmation.
```

A failure priority can be changed only from HIGH to LOW and from LOW to HIGH. It is an error to change the priority level of CRITICAL. (One reason why you may want to change a failure from HIGH to LOW is to avoid seeing it on the default output list of the LIST FAILURE command. For example, if a block corruption has HIGH priority, you may want to temporarily change it to LOW if the block is in a little-used tablespace.)

Open failures are closed implicitly when a failure is repaired. However, you can also explicitly close a failure. This involves a re-evaluation of all other open failures, because some of them may become irrelevant as the result of the closure of the failure.

By default, the command asks the user to confirm a requested change.

Data Recovery Advisor Views

Querying `V$` views:

- `V$IR_FAILURE`: List of all failures, including closed ones (result of the `LIST FAILURE` command)
- `V$IR_MANUAL_CHECKLIST`: List of manual advice (result of the `ADVISE FAILURE` command)
- `V$IR_REPAIR`: List of repairs (result of the `ADVISE FAILURE` command)
- `V$IR_FAILURE_SET`: Cross-reference of failure and advice identifiers



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

See *Oracle Database Reference* for details of the dynamic views that the Data Recovery Advisor uses.

Summary

In this lesson, you should have learned how to:

- Detect and repair database corruption
- Use the Automatic Diagnostic Repository
- Analyze instance recovery with ADRCI
- Use Data Recovery Advisor



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Diagnosing and Repairing Database Failure



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Performing Complete Recovery

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Perform the appropriate type of restore and recovery operation based on the nature of your database failure
- Recover from media failures in data files
- Perform complete recovery



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Ensuring Backups Are Available

RMAN Command	Action
RESTORE PREVIEW	RMAN reports the backups and archived redo log files that RMAN uses to restore and recover the database to the specified time.
RESTORE VALIDATE	RMAN determines which backup sets, data file copies, and archived redo log files need to be restored and then validates them.
RECOVER VALIDATE HEADER	Reports and validates the backups that RMAN could use to restore files needed for the recovery



Copyright © 2020, Oracle and/or its affiliates.

Use the following commands to ensure that all required backups are available and to determine whether you need to direct RMAN to use or avoid specific backups:

- **RESTORE PREVIEW:** Reports on the backups and archived redo log files that RMAN can use to restore and recover the database to the specified time. RMAN queries the metadata, but does not actually read the backup files. The output from this command is in the same format as the `LIST BACKUP` command output.
- **RESTORE VALIDATE:** Specifies that RMAN should decide which backup sets, data file copies, and archived redo log files need to be restored and then validate them. No files are restored. For files on both disk and tape, RMAN reads all blocks in the backup piece or image copy. RMAN also validates off-site backups.
- **RECOVER VALIDATE HEADER:** Reports on and validates the backups that RMAN can use to restore files needed for recovery. When you execute the `RECOVER VALIDATE HEADER` command, RMAN performs the same operations as when you specify the `RESTORE PREVIEW` command. However, in addition to listing the files needed for the restoration and recovery, RMAN validates the backup file headers to determine whether the files on disk or in the media management catalog correspond to what is in the metadata in the RMAN repository.

Restoring in NOARCHIVELOG Mode

If the database is in NOARCHIVELOG mode and if any data file is lost, perform the following tasks:

1. Shut down the instance if it is not already down.
2. Restore the entire database, including all data and control files from a backup.
3. Start the instance and open the database (CDB and all PDBs).
4. Inform users that they must reenter all changes that were made since the last backup.

If the database is in NOARCHIVELOG mode and an incremental backup strategy is implemented, RMAN restores the most recent level 0 backup and then RMAN recovery applies the incremental backups.



Copyright © 2020, Oracle and/or its affiliates.

The loss of *any* data file from a database in NOARCHIVELOG mode requires complete restoration of the database, including control files and all data files.

With the database in NOARCHIVELOG mode, recovery is possible only up to the time of the last backup. So users must reenter all changes made since that backup.

To perform this type of recovery:

1. Shut down the instance if it is not already down.
2. Restore the entire CDB, including all data and controlfiles from the last backup.
3. Start the instance and open the root and PDBs.

If you are using Enterprise Manager Cloud Control, select Whole Database as the recovery type.

If you have a database in NOARCHIVELOG mode that has an incremental backup strategy, RMAN first restores the most recent level 0 and then RMAN recovery applies the incremental backups.

Recovery with Incremental Backups in NOARCHIVELOG Mode

Use incremental backups to perform limited recovery of a database in NOARCHIVELOG mode.

```
STARTUP FORCE NOMOUNT;
RESTORE CONTROLFILE;
ALTER DATABASE MOUNT;
RESTORE DATABASE;
RECOVER DATABASE NOREDO;
ALTER DATABASE OPEN RESETLOGS;
```



Copyright © 2020, Oracle and/or its affiliates.

You can perform limited recovery of a NOARCHIVELOG mode database by using incremental backups. The incremental backups must be consistent backups.

If you have taken incremental backups, RMAN uses your level 0 and level 1 backups to restore and recover the database.

You must specify the NOREDO option on the RECOVER DATABASE command if the online redo log files are lost or if the redo cannot be applied to the incremental backups. If you do not specify the NOREDO option, RMAN searches for the online redo log files after applying the incremental backups. If the online redo log files are not available, RMAN issues an error message.

If the current online redo log files contain all changes since the last incremental backup, you can issue the RECOVER DATABASE command without the NOREDO option, and the changes will be applied.

Note: You need to restore the control file only if it is not current.

Performing Complete Recovery

Loss of a noncritical data file in ARCHIVELOG mode:

- If a data file is lost or corrupted, and if that file does not belong to the SYSTEM or UNDO tablespace, you restore and recover the missing data file while the database is [open](#).
- Recovery is possible up to the time of the last commit, and users are not required to reenter any data.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

With the database in ARCHIVELOG mode, the loss of any data file not belonging to the SYSTEM or UNDO tablespaces affects only the objects that are in the missing file. The rest of the database remains available for users to continue work.

To restore and recover the missing data file:

1. In Cloud Control, navigate to the Perform Recovery page.
2. Select Datafiles as the recovery type and then select “Recover to current time.”
3. Add all data files that need recovery.
4. Determine whether you want to restore the files to the default location or (if a disk or controller is missing) to a new location.
5. Submit the RMAN job to restore and recover the missing files.

Because the database is in ARCHIVELOG mode, recovery is possible up to the time of the last commit, and users are not required to reenter any data.

Performing Complete Recovery

Loss of a critical data file in ARCHIVELOG mode:

1. The instance may or may not shut down automatically. If it does not, use SHUTDOWN ABORT to bring the instance down.
2. Mount the database.
3. Restore and recover the missing data file.
4. Open the database.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Data files belonging to the SYSTEM tablespace or containing UNDO data are considered system critical. A loss of one of these files requires the database to be restored from the MOUNT state (unlike other data files that may be restored with the database open).

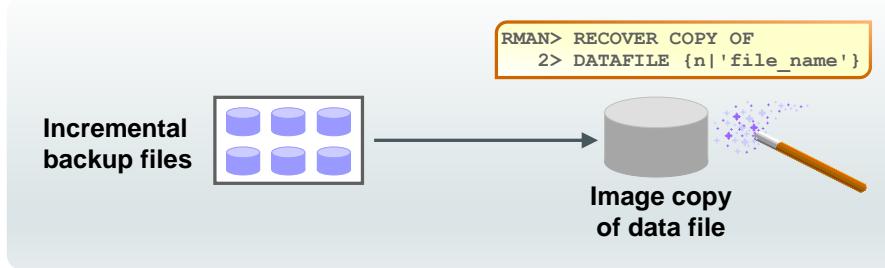
To perform this recovery:

1. If the instance is not already shut down, shut it down.
2. Mount the database.
3. In Enterprise Manager Cloud Control, navigate to the Perform Recovery page.
4. Select Datafiles as the recovery type and then select “Recover to current time.”
5. Add all data files that need recovery.
6. Determine whether you want to restore the files to the default location or (if a disk or controller is missing) to a new location.
7. Submit the RMAN job to restore and recover the missing files.
8. Open the database. Users are not required to re-enter data because the recovery is up to the time of the last commit.

Review: Recovering Image Copies

RMAN can recover image copies by using incremental backups:

- Image copies are updated with all changes up to the incremental backup SCN.
- Incremental backup reduces the time required for media recovery.
- There is no need to perform an image copy after the incremental restoration.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

You can use RMAN to apply incremental backups to data file image copies. With this recovery method, you use RMAN to recover a copy of a data file—that is, you roll forward (recover) the image copy to the specified point in time by applying the incremental backups to the image copy. The image copy is updated with all changes up through the SCN at which the incremental backup was taken. RMAN uses the resulting updated data file in media recovery just as it would use a full image copy taken at that SCN, without the overhead of performing a full image copy of the database every day. The following are the benefits of applying incremental backups to data file image copies:

- You reduce the time required for media recovery (using archive logs) because you need to apply archive logs only since the last incremental backup.
- You do not need to perform a full image copy after the incremental restoration.

If the recovery process fails during the application of the incremental backup file, you simply restart the recovery process. RMAN automatically determines the required incremental backup files to apply, from before the image data file copy until the time at which you want to stop the recovery process. If there is more than one version of an image copy recorded in the RMAN catalog, RMAN automatically uses the latest version of the image copy. RMAN reports an error if it cannot merge an incremental backup file with an image copy.

Recovering Image Copies: Example

If you run these commands daily:

```
RMAN> recover copy of database with tag 'daily_inc';
RMAN> backup incremental level 1 for recover of copy
2> with tag 'daily_inc' database;
```

This is the result:

	RECOVER	BACKUP
Day 1	Nothing	Create image copies
Day 2	Nothing	Create incremental level 1
Day 3 and onward	Recover copies based on incremental	Create incremental level 1



Copyright © 2020, Oracle and/or its affiliates.

If you run the commands shown in the slide daily, you get continuously updated image copies of all the database data files at any time.

The chart shows what happens for each run. Note that this algorithm requires some priming; the strategy does not come to fruition until after day 3.

- **Day 1:** The RECOVER command does nothing. There exist no image copies to recover yet. The BACKUP command creates the image copies.
- **Day 2:** The RECOVER command, again, does nothing. This is because there is no incremental backup yet. The BACKUP command creates the incremental backup, now that baseline image copies have been created on day 1.
- **Day 3:** The RECOVER command applies the changes from the incremental backup to the image copies. The BACKUP command takes another incremental backup, which will be used to recover the image copies on day 4. The cycle continues like this.

It is important to use tags when implementing this kind of backup strategy. They serve to link these particular incremental backups to the image copies that are made. Without the tag, the most recent, and possibly incorrect, incremental backup would be used to recover the image copies.

Performing a Fast Switch to Image Copies

Perform fast recovery by performing the following steps:

1. Take data files offline.
2. Use the SWITCH TO ... COPY command to switch to image copies.
3. Recover data files.
4. Bring data files online.

→ Optionally, do the following to put the files back into their original location:

5. Create an image copy of the data file in the original location.
6. Take data files offline.
7. SWITCH TO ... COPY
8. Recover data files.
9. Bring data files online.

Now the data files are recovered and usable in their new location.

```
SQL> SWITCH DATAFILE 'filename' TO COPY;
```



Copyright © 2020, Oracle and/or its affiliates.

You can use image copies of data files for fast recovery by performing the following steps:

1. Take the data file offline.
2. Use the SWITCH TO ... COPY command to point to the image copy of the files.
3. Recover the data files.
4. Bring the data files online.

At this point, the database is usable, and the data files are recovered. But if you want to put the data files back into their original location, proceed with the following steps:

5. Create an image copy of the data files in the original location by using the BACKUP AS COPY command.
6. Take the data files offline.
7. Switch to the copy you made in step 5 by using the SWITCH TO COPY command.
8. Recover the data files.
9. Bring the data files online.

You can recover data files, tablespaces, tempfiles, or the entire database with this command. The files being switched to must be image copies.

Using SET NEWNAME for Switching Files

- Use the SET NEWNAME command in a RUN block to restore to a nondefault location.

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev2 DEVICE TYPE sbt;
  SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
  SET NEWNAME FOR DATAFILE '/disk1/oradata/prod/users01.dbf'
    TO '/disk2/users01.dbf';
  RESTORE TABLESPACE users;
  SWITCH DATAFILE ALL;
  RECOVER TABLESPACE users;
  SQL "ALTER TABLESPACE users ONLINE";
}
```

- Instead of individual names, specify a default name format for all files in a database or in a named tablespace.
- The default name is used for DUPLICATE, RESTORE, and SWITCH commands in the RUN block.



Copyright © 2020, Oracle and/or its affiliates.

The SET NEWNAME command can be used only inside a RUN block. It prepares a name mapping for subsequent operations. In the example in the slide, the SET NEWNAME command defines the location where a restore operation of that data file will be written. When the RESTORE command executes, the users01.dbf data file is restored to /disk2/users01.dbf. It is written there, but the control file is still not pointing to that location. The SWITCH command causes the control file to be updated with the new location.

A more efficient way is to use the SET NEWNAME clause to specify the default name format for all data files in a named tablespace and all data files in the database.

The order of precedence for the SET NEWNAME command is as follows:

1. SET NEWNAME FOR DATAFILE and SET NEWNAME FOR TEMPFILE
2. SET NEWNAME FOR TABLESPACE
3. SET NEWNAME FOR DATABASE

Using Restore Points

A restore point provides a name to a point in time:

- Now:

```
SQL> CREATE RESTORE POINT before_mods;
```

- Some time in the past:

```
SQL> CREATE RESTORE POINT end_q1 AS OF SCN 100;
```

Timeline



Copyright © 2020, Oracle and/or its affiliates.

A restore point is a user-defined name associated with an SCN of the database corresponding to the time of the creation of the restore point. There are two types of restore points:

- Guaranteed restore point:** Enforces the retention of flashback logs required for Flashback Database back to any point in time after the creation of the earliest guaranteed restore point. A guaranteed restore point does not age out of the control file and must be explicitly dropped.
- Normal restore point:** A label for an SCN or time. Normal restore points exist in a circular list and can be overwritten in the control file.

You can give a name to a particular point in time or an SCN number. This is useful for future reference, when performing point-in-time recovery or flashback operations.

- The first example in the slide creates a restore point that represents the present point in time. If you were about to apply an update of an application or data in the database, and you wanted to refer back to this state of the database, you could use the BEFORE_MODS restore point.
- The second example in the slide creates a restore point representing a past SCN, 100. This restore point can be used in the same ways as the previous one.

Normally, restore points are maintained in the database for at least as long as specified by the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter. However, you can use the PRESERVE option when creating a restore point, which causes the restore point to be saved until you explicitly delete it.

You can see restore points in the V\$RESTORE_POINT view with name, SCN, time stamp, and other information.

PDB Tempfile Recovery

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

```
SQL> CONNECT local_user@HR_PDB
SQL> select * from my_table order by 1,2,3,4,5,6,7,8,9,10,11,12,13;
      select * from my_table order by 1,2,3,4,5,6,7,8,9,10,11,12,13
      *
ERROR at line 1:
ORA-01565: error in identifying file
  '/u01/app/oracle/oradata/CDB1/HR_PDB/temp2_01.dbf'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
```

- Temporary files are automatically re-created at PDB startup.
- Manual re-creation is also possible.



Copyright © 2020, Oracle and/or its affiliates.

If a tempfile belonging to a PDB temporary tablespace is lost or damaged, and the user issuing the statement uses it, an error occurs during the execution of SQL statements that require that temporary space for sorting.

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

The PDB can open with a missing temporary file. If any of the temporary files do not exist when the PDB is opened, they are re-created automatically at PDB startup.

You can perform a manual re-creation instead, while connected to the PDB:

```
SQL> ALTER TABLESPACE temp ADD TEMPFILE
      '/u01/app/oracle/oradata/CDB1/HR_PDB/temp2_02.dbf'
      SIZE 20M;
SQL> ALTER TABLESPACE temp DROP TEMPFILE
      '/u01/app/oracle/oradata/CDB1/HR_PDB/temp2_01.dbf';
```

PDB SYSTEM or UNDO Tablespace Recovery

- The CDB and all other PDBs can be left open.
- Perform these steps:
 1. Connect to the PDB.
 2. Shut down the PDB with the ABORT option if it is not automatically done.

```
$ sqlplus sys@sales_pdb as sysdba  
SQL> SHUTDOWN ABORT
```

```
SQL> ALTER PLUGGABLE DATABASE CLOSE ABORT;
```

3. Restore and recover the PDB or the missing tablespace or the damaged datafile.

```
$ rman target sys@sales_pdb  
RMAN> RESTORE DATABASE;  
RMAN> RECOVER DATABASE;  
RMAN> ALTER PLUGGABLE DATABASE sales_pdb OPEN;
```



Copyright © 2020, Oracle and/or its affiliates.

Closing a PDB with the ABORT mode forcefully closes it without bringing down the entire CDB instance. It is successful regardless of the states of the actual PDB datafiles.

This operation requires you to be connected to the PDB, have the ALTER PLUGGABLE DATABASE system privilege, and have the CDB in ARCHIVELOG mode.

If the datafile that is missing or corrupted belongs to a PDB and more specifically to the SYSTEM or UNDO tablespace, the PDB must be shut down in ABORT mode.

The health checker can automatically detect that there is a severe failure and aborts the PDB.

```
Checker run found 1 new persistent data failures  
2015-10-07T01:17:39.950612+00:00  
SALES_PDB(5) :ALTER PLUGGABLE DATABASE CLOSE ABORT  
2015-10-07T01:17:39.978484+00:00  
SALES_PDB(5) :KILL SESSION for sid=(254, 7528) :  
SALES_PDB(5) : Reason = PDB close abort  
SALES_PDB(5) : Mode = KILL HARD FORCE -/-/-
```

A PDB, tablespace, or datafile media recovery is required before the PDB can be reopened.

In case the PDB is the application root of an application container, other datafiles in the application PDBs may have to be restored and recovered as well.

PDB Non-SYSTEM Tablespace Recovery

- It is similar to recovery in non-CDBs.
- Perform the recovery within the PDB.
 1. Connect to the PDB.
 2. Take the tablespace offline.
- Other PDBs are not impacted.
- You can also use the REPAIR command.

```
SQL> CONNECT system@sales_pdb
SQL> ALTER TABLESPACE tbs2 OFFLINE IMMEDIATE;
RMAN> CONNECT TARGET /
RMAN> RESTORE TABLESPACE sales_pdb:tbs2;
RMAN> RECOVER TABLESPACE sales_pdb:tbs2;

SQL> ALTER TABLESPACE tbs2 ONLINE;
```



Copyright © 2020, Oracle and/or its affiliates.

If the datafile that is missing or corrupted belongs to a PDB and more specifically to any tablespace other than SYSTEM tablespace, the PDB need not be closed. The datafile with the error is taken OFFLINE IMMEDIATE. Media recovery for that datafile is required and performed in a similar way to media recovery of a set of tablespaces. During that recovery time, users can work with other PDB tablespaces and within other PDBs.

RMAN RESTORE and RECOVER commands are frequently run consecutively. The RMAN REPAIR command accepts DATAFILE, TABLESPACE, PLUGGABLE DATABASE, and DATABASE options and performs both RESTORE and RECOVER commands. Where possible, REPAIR automatically takes a file offline, restores and recovers it, and then brings it back online again:

1. Automatically takes files offline (if applicable)
2. Restores designated files
3. Recovers designated files
4. Automatically brings back online any files taken offline in step 1

Summary

In this lesson, you should have learned how to:

- Perform the appropriate type of restore and recovery operation based on the nature of your database failure
- Recover from media failures in data files
- Perform complete recovery



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Recovering from the Loss of a System-Critical Data File
- Recovering from the Loss of an Application Data File



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Performing Point-in-Time Recovery

The Oracle logo, consisting of the word "ORACLE" in white capital letters inside a red rectangular box.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Distinguish and describe point-in-time recovery (PITR) of the database, tablespace, and table
- Identify the circumstances where PITR is a good solution and where it cannot be used
- List what operations occur when you perform a point-in-time recovery
- Determine the correct target time for the point-in-time recovery
- Perform automated TSPITR
- Perform table recovery from backups



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Point-in-Time Recovery

Point-in-time recovery benefits:

- Quick recovery of one or more objects to an earlier time
- No effect on other objects

Recovery scope:

- Database point-in-time recovery (DBPITR), also referred to as incomplete recovery
- Tablespace point-in-time recovery (TSPITR)
- Table point-in-time recovery (TPITR)



Copyright © 2020, Oracle and/or its affiliates.

A benefit of PITR is that you can recover one or more objects—for example, tablespaces—to an earlier time, without affecting the state of the other tablespaces and objects in the database.

Point-in-time recovery has three different levels or recovery scopes:

- Database point-in-time recovery (DBPITR):
 - To recover of an entire database to a specified past target time, SCN, or log sequence number
 - To migrate a database to a different platform by creating a new database on the destination platform and performing a transport of all the user tablespaces, but excluding the SYSTEM tablespace
- Tablespace point-in-time recovery (TSPITR):
 - To recover one or more contained tablespaces to an earlier point in time
- Table point-in-time recovery (TPITR):
 - To recover one or more tables or table partitions to an earlier point in time

PITR Terminology

- **Target time:** The point in time or SCN that an object will be recovered to
- **Recovery set:** For tablespaces, the data files to be recovered
- **Auxiliary set:** Required data files that are not part of the recovery set. It typically includes:
 - SYSTEM tablespace
 - Undo segment tablespaces
 - Temporary tablespace
- **Auxiliary destination:** Disk location to store files



Copyright © 2020, Oracle and/or its affiliates.

The following terminology is used when discussing PITR:

- **Target time:** The point in time or system change number (SCN) that the object will be recovered to
- **Recovery set:** For example, data files composing the tablespaces to be recovered
- **Auxiliary set:** Required data files (for metadata and dependencies) that are not themselves part of the recovery set. The auxiliary set typically includes:
 - A copy of the SYSTEM tablespace
 - Data files that contain undo segments from the target instance
 - In some cases, a temporary tablespace, used during the export of database objects from the auxiliary instance
- **Auxiliary destination:** A location on disk that can be used to store any of the auxiliary set data files, control files, and online logs of the auxiliary instance during PITR. Files stored in the auxiliary destination can be deleted after PITR is complete.

Performing Point-in-Time Recovery

Perform database point-in-time (incomplete) recovery by doing the following:

1. Determine the target point of the restore: SCN, time, restore point, or log sequence number.
2. Set the NLS environment variables appropriately.
3. Mount the database.
4. Prepare and execute a RUN block, using the SET UNTIL, RESTORE, and RECOVER commands.
5. Open the database in READ ONLY mode and verify that the recovery point is correct.
6. Open the database by using RESETLOGS.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

You can perform server-managed point-in-time recovery by using the following steps. The database must be in ARCHIVELOG mode.

1. Determine the restore target. This can be in terms of a date and time, an SCN, restore point, or log sequence number. For example, if you know that some bad transactions were submitted at 3:00 PM yesterday, then you can choose 2:59 PM yesterday as the target restore point time.
2. Set the National Language Support (NLS) OS environment variables so that the time constants you provide to RMAN are formatted correctly. These are some example settings:

```
$ export NLS_LANG = american_america.us7ascii  
$ export NLS_DATE_FORMAT = "yyyy-mm-dd:hh24:mi:ss"
```

3. Mount the database. If it is open, you have to shut it down first, as in this example:

```
RMAN> shutdown immediate  
RMAN> startup mount
```

4. Create a RUN block and run it. The RECOVER and RESTORE commands should be in the same RUN block so that the UNTIL setting applies to both. For example, if you choose to recover to a particular SCN, the RESTORE command needs to know that value so it restores files from backups that are sufficiently old—that is, backups that are from before that SCN. Here is an example of a RUN block:

```
RUN
{
  SET UNTIL TIME '2018-08-14:21:59:00';
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

Note: If you do not want to rely on the NLS parameters, you can set the time explicitly, for example, in a European format:

```
set until time
"to_date('14.08.2018 21:59:00','dd.mm.yyyy hh24:mi:ss'))";
```

5. As soon as you open the database for read/write, you have committed to the restore you just performed. So, first, open the database READ ONLY, and view some data, to check whether the recovery did what you expected.

```
RMAN> SQL 'ALTER DATABASE OPEN READ ONLY';
```

6. If satisfied with the results of the recovery, open the database with the RESETLOGS option, as shown:

```
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

When to Use TSPITR

TSPITR can be used in the following situations:

- To recover data lost after an erroneous TRUNCATE TABLE statement
- To recover from logical corruption of a table
- To undo the effects of a batch job or DML statements that have affected only a part of the database
- To recover a logical schema to a different point from the rest of the physical database
- To recover a dropped tablespace
- Can be performed repeatedly to points-in-time before the tablespace was brought online without requiring a recovery catalog



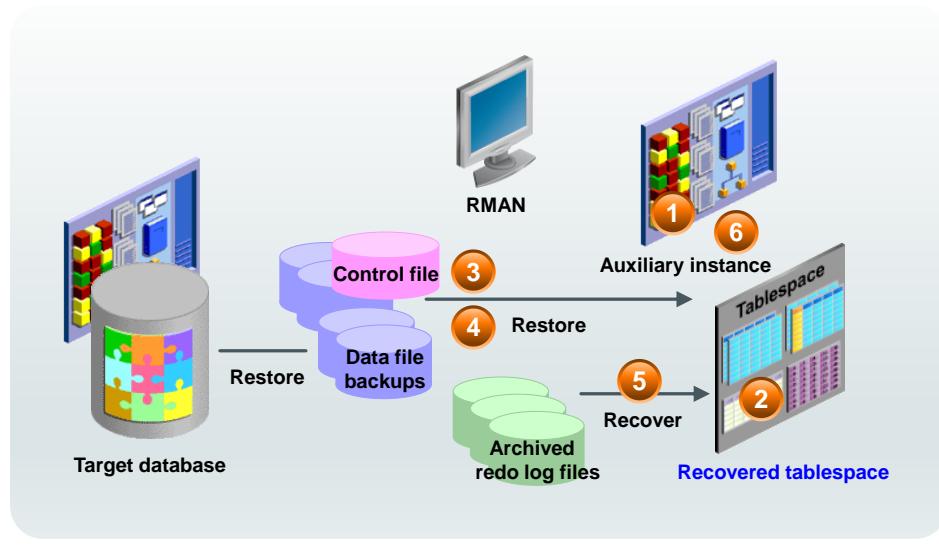
Copyright © 2020, Oracle and/or its affiliates.

RMAN TSPITR can be used to:

- Recover data lost after an erroneous TRUNCATE TABLE statement
- Recover from logical corruption of a table
- Undo the effects of an incorrect batch job or another data manipulation language (DML) statement that has affected only a subset of the database
- Recover a logical schema to a different point in time than other parts of the physical database

TSPITR uses transportable tablespaces and Data Pump. Because of this technology, TSPITR can be used to recover a dropped tablespace. In addition, TSPITR can be performed repeatedly to different points in time without the need for a recovery catalog.

Tablespace Point-in-Time Recovery: Architecture



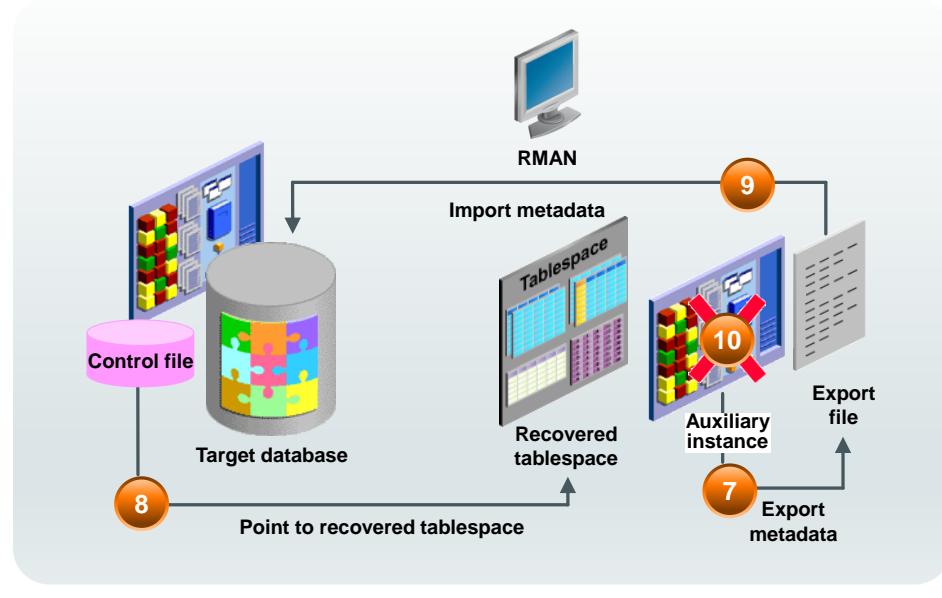
In the diagram, the following TSPITR entities are shown:

- Target database:** Contains the tablespace to be recovered
- Control file:** Provides backup information to RMAN
- Backup sets:** Come from the target database and are the source of the reconstructed tablespace
- Archived redo logs:** Come from the target database and are the source of the reconstructed tablespace
- Auxiliary instance:** Is the Oracle database instance used during the recovery process to perform the recovery

RMAN performs the following steps:

- Creates the auxiliary instance, starts it, and connects to it
- Takes the tablespaces that will be recovered offline
- Restores a backup control file from a point in time before the target time to the auxiliary instance
- Restores the data files from the recovery set and the auxiliary set to the auxiliary instance
- Recovers the restored data files to the specified time
- Opens the auxiliary database with the `RESETLOGS` option

Tablespace Point-in-Time Recovery: Architecture



7. Exports the dictionary metadata about objects in the recovered tablespaces to the target database
8. Shuts down the auxiliary instance. (Issues `SWITCH` commands on the target database so that the target database control file points to the data files in the recovery set that were recovered on the auxiliary instance.)
9. Imports the dictionary metadata from the auxiliary instance to the target instance
10. Deletes all auxiliary set files

Preparing for TSPITR

To prepare for TSPITR, perform the following steps:

- Determine the correct target time.
- Determine what is needed in the recovery set.
- Identify and preserve objects that will be lost after PITR.



Copyright © 2020, Oracle and/or its affiliates.

Before performing TSPITR, you need to determine the correct target time for your recovery. You need to determine whether you need additional tablespaces in your recovery set. You should evaluate what objects will be lost as a result of the TSPITR operation and determine how you want to preserve those objects.

Each of these steps is discussed in more detail in this lesson.

Determining the Correct Target Time

- After you perform TSPITR and bring the tablespace online, you cannot use a backup from an earlier time (unless you are using a recovery catalog).
- Use the following methods to determine the correct target time:
 - Flashback Query
 - Flashback Transaction Query
 - Flashback Version Query
- Simple alternative to TSPITR: Flashback data (if still available as undo)



Copyright © 2020, Oracle and/or its affiliates.

It is extremely important that you choose the right target time or SCN for TSPITR. After you perform TSPITR and bring a tablespace online, you cannot use any backup from a time earlier than the moment you brought the tablespace online. In practice, this means that you cannot make a second attempt at TSPITR if you choose the wrong target time the first time, unless you are using a recovery catalog. However, if you have a recovery catalog, you can perform repeated TSPITR operations to different target times.

The current control file does not contain a record of an older incarnation of the recovered tablespace if you do not use a recovery catalog. Recovery with a current control file that involves the tablespace cannot use a backup taken prior to the time when you brought the tablespace online. However, you can perform incomplete recovery of the whole database to any time prior to or equal to the time when you brought the tablespace online if you can restore a backup control file from before that time.

You can use Oracle Flashback Query, Oracle Flashback Transaction Query, and Oracle Flashback Version Query to investigate changes to your database and to help determine the correct target time for TSPITR.

Note: With the Flashback tools and the data still available as undo data, it is usually much simpler to use the Flashback tools for undoing unwanted changes (rather than TSPITR).

Determining the Tablespaces for the Recovery Set

- If objects in the tablespace that you are recovering have relationships with objects in other tablespaces, you can:
 - Add the tablespace that contains the related objects to the recovery set
 - Suspend the relationship for the duration of TSPITR
 - Remove the relationship
- Use the DBMS_TTS.TRANSPORT_SET_CHECK procedure to determine whether the tablespaces in the recovery set are self-contained.

```
DBMS_TTS.TRANSPORT_SET_CHECK ('USERS,EXAMPLE');
SELECT * FROM TRANSPORT_SET_VIOLATIONS;
```



Copyright © 2020, Oracle and/or its affiliates.

To identify relationships between objects that span the recovery set boundaries, use the DBMS_TTS.TRANSPORT_SET_CHECK procedure and query the TRANSPORT_SET_VIOLATIONS view.

Note: RMAN TSPITR automatically executes the DBMS_TTS.TRANSPORT_SET_CHECK procedure for the recovery set tablespaces and verifies that the query against TRANSPORT_SET_VIOLATIONS returns no rows. If the query returns rows, RMAN stops TSPITR processing, and any tablespace containment violations must be resolved before TSPITR can proceed. You can execute the procedure and query the view as described in the slide as a precautionary measure.

Identifying Objects That Will Be Lost

- Objects created in the tablespace after the target recovery time are lost.
- Query `TS_PITR_OBJECTS_TO_BE_DROPPED` to determine which objects will be lost after TSPTR.
- Use Export before TSPTR and Import after TSPTIR to preserve and re-create the lost objects.



Copyright © 2020, Oracle and/or its affiliates.

Query the `TS_PITR_OBJECTS_TO_BE_DROPPED` view to determine whether there are any objects that will be lost as a result of performing tablespace point-in-time recovery.

As an example, you are performing TSPTR for the `USERS` and `EXAMPLE` tablespaces to the target time of April 3, 2018, at 8:30:00 AM. Issue the following query to determine whether there are any objects that will be lost after your TSPTR:

```
SELECT OWNER, NAME, TABLESPACE_NAME,  
TO_CHAR(CREATION_TIME, 'YYYY-MM-DD:HH24:MI:SS')  
FROM TS_PITR_OBJECTS_TO_BE_DROPPED  
WHERE TABLESPACE_NAME IN ('USERS', 'EXAMPLE')  
AND CREATION_TIME >  
    TO_DATE('2018-APR-03:08:30:00', 'YY-MON-DD:HH24:MI:SS')  
ORDER BY TABLESPACE_NAME, CREATION_TIME;
```

Performing RMAN TSPITR

- Fully automated TSPITR:
 - Specify an auxiliary destination.
 - RMAN manages all aspects of TSPITR.
 - This is the recommended method.
- Customized TSPITR with an automatic auxiliary instance:
 - This is based on fully automated TSPITR.
 - Customize the location of files.
 - Specify initialization parameters.
 - Specify channel configurations.
- TSPITR using your own auxiliary instance:
 - Configure and manage the auxiliary instance.



Copyright © 2020, Oracle and/or its affiliates.

You have the following options when performing TSPITR:

- **Fully automated TSPITR:** Specify an auxiliary destination, and RMAN manages all aspects of the TSPITR operation. This is the simplest way to perform TSPITR and is recommended unless you specifically need more control over the location of recovery set files after TSPITR, or auxiliary set files during TSPITR, or control over the channel configurations, or some other aspect of your auxiliary instance.
- **Customized TSPITR with an automatic auxiliary instance:** TSPITR is based on the behavior of fully automated TSPITR, possibly still using an auxiliary destination. You can customize one or more aspects of the behavior, such as the location of auxiliary set or recovery set files. You can specify initialization parameters or channel configurations for the auxiliary instance created and managed by RMAN.
- **TSPITR with your own auxiliary instance:** Set up, start, stop, and clean up the auxiliary instance used in TSPITR. In addition, you can manage the TSPITR process by using some of the methods available in customized TSPITR with an automatic auxiliary instance.

Performing Fully Automated TSPITR

1. Configure channels required for TSPITR on the target instance.
2. Specify the auxiliary destination by using the AUXILIARY DESTINATION option.

```
RMAN> CONNECT TARGET
RMAN> RECOVER TABLESPACE users, example
> UNTIL TIME '2018-06-29:08:00:00'
> AUXILIARY DESTINATION
> '/u01/app/oracle/oradata/aux';
```

3. Back up the recovered tablespaces and bring them online.



Copyright © 2020, Oracle and/or its affiliates.

In addition to the preparation requirements discussed earlier in the lesson, when you perform fully automated TSPITR, you must:

- Configure any channels required for TSPITR on the target instance
- Specify a destination for RMAN to use for the auxiliary set of data files and other auxiliary instance files

After TSPITR has completed, back up the recovered tablespaces and bring them online. You cannot use backups of any tablespaces that participate in TSPITR taken before TSPITR after you perform TSPITR.

Note: This time format assumes that NLS_DATE_FORMAT is set to 'yyyy-mm-dd:hh24:mi:ss' and NLS_LANG is set to AMERICAN_AMERICA.WE8MSWIN1252.

Improving TSPITR Performance

- `CONFIGURE AUXNAME` for a persistent alternative location for an auxiliary set data file image copy
- `SET NEWNAME` for an alternative location for the duration of a `RUN` command

```
RUN
{
  SET NEWNAME FOR DATAFILE '$ORACLE_BASE/oradata/orcl/users01.dbf'
  TO '/u01/backup/users01.dbf';

  RECOVER TABLESPACE users UNTIL SEQUENCE 1300 THREAD 1;
}
```



Copyright © 2020, Oracle and/or its affiliates.

You can improve TSPITR performance by directing RMAN to use the existing image copies of the recovery set and auxiliary set data files. This technique enables RMAN to skip restoring the data files from a backup.

The `CONFIGURE AUXNAME` command sets a persistent alternative location for an auxiliary set data file image copy, whereas the `SET NEWNAME` command sets an alternative location for the duration of a `RUN` command.

Performing RMAN TSPITR with an RMAN-Managed Auxiliary Instance

- Rename or relocate your recovery set data files.
- Specify a location other than the auxiliary destination for some or all of the auxiliary set data files.
- Create image copy backups of your data files before TSPITR.
- Use a different channel configuration for the auxiliary instance.
- Specify different initialization parameters for your RMAN-managed auxiliary instance.



Copyright © 2020, Oracle and/or its affiliates.

If you want to customize RMAN TSPITR, you can use an RMAN-managed auxiliary instance and make the following changes:

- Rename the recovery set data files by using `SET NEWNAME` so that they are not restored and recovered in their original locations.
- Control the location of your auxiliary set data files by specifying new names for individual files with `SET NEWNAME` and using `DB_FILE_NAME_CONVERT` to provide rules for converting data file names in the target database to data file names for the auxiliary database.
- Use existing image copies of the recovery set and auxiliary set data files on disk rather than restoring them from backup for faster RMAN TSPITR performance.

Note: Refer to *Oracle Database Backup and Recovery User's Guide* for additional information.

Performing RMAN TSPITR by Using Your Own Auxiliary Instance

- Not recommended, but supported
- Perform the following steps:
 1. Create an Oracle password file for the auxiliary instance.
 2. Create an initialization parameter file for the auxiliary instance.
 3. Verify Oracle Net connectivity to the auxiliary instance.
 4. Start the auxiliary instance in `NOMOUNT` mode.
 5. Connect the RMAN client to the target and auxiliary instances.
 6. Execute the `RECOVER TABLESPACE` command.



Copyright © 2020, Oracle and/or its affiliates.

Oracle recommends that you allow RMAN to manage the creation and destruction of the auxiliary instance used during RMAN TSPITR. However, creating and using your own auxiliary instance is supported.

To create an Oracle instance suitable for use as an auxiliary instance, perform the following steps:

1. Create an Oracle password file for the auxiliary instance by using the `orapwd` utility.
2. Create a text initialization parameter file for the auxiliary instance.
3. Verify Oracle Net connectivity to the auxiliary instance by using a valid net service name.

To perform TSPITR, perform the following steps:

4. Start the auxiliary instance in `NOMOUNT` mode.
5. Connect the RMAN client to target and auxiliary instances.
6. Execute the `RECOVER TABLESPACE` command.

Refer to *Oracle Database Backup and Recovery User's Guide* for a detailed example.

Troubleshooting RMAN TSPITR

- File name conflicts: Ensure that there are no name conflicts when using `SET NEWNAME`, `CONFIGURE AUXNAME`, and `DB_FILE_NAME_CONVERT`.
- RMAN cannot identify tablespaces with undo segments: Use the `UNDO TABLESPACE` clause.
- Restarting a manual auxiliary instance after TSPITR failure: Shut down and restart in `NOMOUNT` mode.



Copyright © 2020, Oracle and/or its affiliates.

File name conflicts: If your use of `SET NEWNAME`, `CONFIGURE AUXNAME`, and `DB_FILE_NAME_CONVERT` causes multiple files in the auxiliary or recovery sets to have the same name, you receive an error during TSPITR. To correct the problem, specify different values for these parameters to eliminate the duplicate name.

RMAN cannot identify tablespaces with undo segments: During TSPITR, RMAN needs information about which tablespaces had undo segments at the TSPITR target time. This information is usually available in the recovery catalog, if one is used. If there is no recovery catalog, or if the information is not found in the recovery catalog, RMAN proceeds, assuming that the set of tablespaces with undo segments at the target time is the same as the set of tablespaces with undo segments at the present time. If this assumption is not correct, the TSPITR operation fails and an error is reported. To prevent this from happening, provide a list of tablespaces with undo segments at the target time in the `UNDO TABLESPACE` clause.

Restarting manual auxiliary instance after TSPITR failure: If you are managing your own auxiliary instance and there is a failure in TSPITR, then before you can retry TSPITR, you must shut down the auxiliary instance, correct the problem, and put the auxiliary instance back in `NOMOUNT` mode.

PITR of PDBs

- PDB PITR

```
RMAN> ALTER PLUGGABLE DATABASE pdb1 CLOSE;
RMAN> RUN {
      SET UNTIL SCN = 1851648 ;
      RESTORE pluggable DATABASE pdb1;
      RECOVER pluggable DATABASE pdb1
          AUXILIARY DESTINATION='/u01/app/oracle/oradata';
      ALTER PLUGGABLE DATABASE pdb1 OPEN RESETLOGS;
    }
```

- PDB TSPITR

```
RMAN> RECOVER TABLESPACE pdb1:test_tbs
      UNTIL SCN 832972
      AUXILIARY DESTINATION '/tmp/CDB1/reco';
RMAN> ALTER TABLESPACE pdb1:test_tbs ONLINE;
```

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

Recovering a PDB to a point in time does not affect all parts of the CDB—the whole CDB is still opened and, therefore, all other PDBs are opened. After recovering a PDB to a specified point in time, when you open the PDB using the `RESETLOGS` option, a new incarnation of the PDB is created. The PDB `RESETLOGS` does not perform a `RESETLOGS` for the CDB.

- A PDB record in the control file is updated.
- Each redo log record carries PDB ID in the redo header. This is how recovery knows which redo applies to which PDB. Redo logs are shared by all PDBs—redo from each PDB is written to a single set of redo logs.

Conceptually, a PDB resetlogs is similar to a database resetlogs.

After recovery, the old backup of the PDB remains valid and can be used if a media failure occurs. A PDB incarnation is a subincarnation of the CDB. For example, if the CDB is incarnation 5, and a PDB is incarnation 3, then the fully specified incarnation number of the PDB is (5, 3). The initial incarnation of a PDB is 0. To view the incarnation of a PDB, query the `V$PDB_INCARNATION` view.

If you do not use a fast recovery area, you must specify the temporary location of the auxiliary set files by using the `AUXILIARY DESTINATION` clause (example in the slide).

Each PDB has its own set of tablespaces. TSPITR can be used to recover a tablespace to an earlier point in time. Perform the TSPITR as described in the second example in the slide, specifying the full tablespace name including the PDB name.

Recovering a tablespace of a PDB to a point in time does not affect all parts of the CDB —the whole CDB is still opened and, therefore, all PDBs are opened. After recovering a tablespace back in time, put the tablespace back online.

Recovering Tables from Backups

When to recover tables and table partitions from RMAN backups:

- Small number of tables (no TSPITR)
- Not in a self-contained tablespace (no TSPITR)
- Purged tables (no Flashback drop)
- Beyond the available undo (no Flashback Table)
- After a structural DDL change (no Flashback Table)



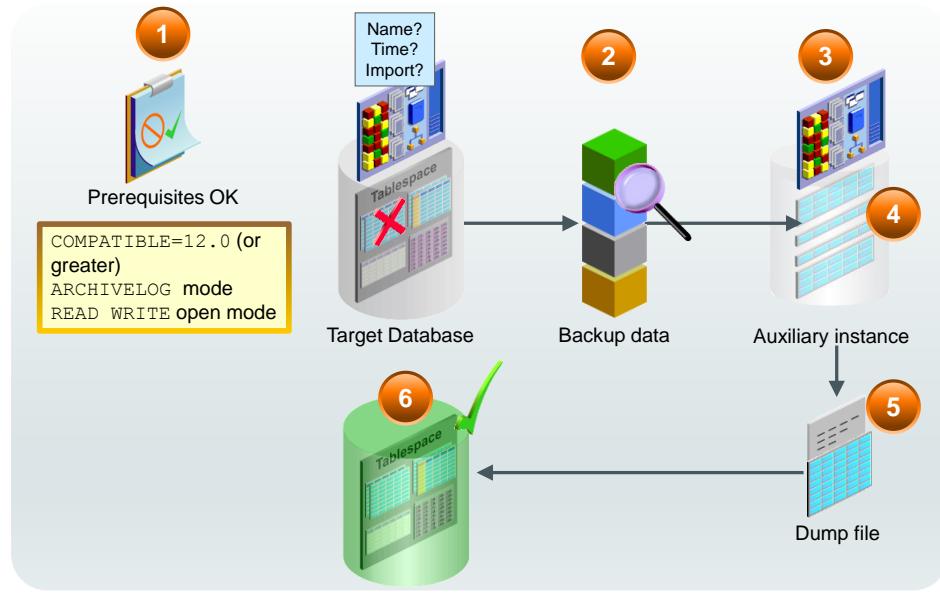
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

RMAN enables you to recover one or more tables or table partitions to a specified point in time without affecting the remaining database objects. Recovering tables and table partitions is useful in the following situations:

- You need to recover a very small number of tables to a particular point in time. In this situation, TSPITR is not the most effective solution because it moves all the objects in the tablespace to a specified point in time.
- You need to recover a tablespace that is not self-contained to a particular point in time. TSPITR can be used only if the tablespace is self-contained.
- You need to recover tables that have been either corrupted or deleted with the PURGE option, so you cannot use the Flashback Drop functionality.
- You enabled logging for a Flashback Table, but the flashback target time or SCN is beyond the available undo.
- You want to recover data that is lost after a data definition language (DDL) operation has changed the structure of tables. You cannot use Flashback Table to rewind a table to before the point of a structural change, such as a truncate table operation.

Table Recovery: Graphical Overview



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

1. RMAN uses backups that were previously created to recover tables and table partitions to a specified point in time. The prerequisites are fulfilled, and you provide the following input with the `RECOVER` command:
 - Names of tables or table partitions to be recovered
 - Point in time to which the tables or table partitions need to be recovered
 - Whether the recovered tables or table partitions must be imported into the target database
 - RMAN uses your input to automate the process of recovering the specified tables or table partitions.
2. RMAN determines the backup based on your specification.
3. RMAN creates an auxiliary instance.
4. RMAN recovers your tables or table partitions, up to the specified point in time, into this auxiliary instance.
5. RMAN creates a Data Pump export dump file that contains the recovered objects.
6. RMAN imports the recovered objects into the target database.

Prerequisites and Limitations

To recover tables and table partitions from RMAN backups, the target database must be:

- In read/write mode
- In ARCHIVELOG mode

Limitations of recovery: No tables and table partitions from:

- The SYS schema
- The SYSTEM and SYSAUX tablespaces
- Standby databases



Copyright © 2020, Oracle and/or its affiliates.

The slide lists the prerequisites and limitations for table recovery from backups.

The result of table recovery is of course that the table exists as it was at an earlier point in time. This could potentially cause consistency issues.

Specifying the Recovery Point in Time

Recover table and table partitions to the state they were in by specifying:

- UNTIL SCN *integer*: The system change number (SCN)
- UNTIL TIME '*date_string*': The time in the date format:
 - Of the NLS_LANG and NLS_DATE_FORMAT environment variables, or
 - Date constants, for example, SYSDATE – 5
- UNTIL SEQUENCE *integer* (THREAD *integer*): The log sequence number and thread number



Copyright © 2020, Oracle and/or its affiliates.

RMAN recovers tables or table partitions to the state that they were in at the time specified by the SCN. The SCN is an upper, noninclusive limit.

RMAN recovers tables or table partitions to the state they were in at the specified time. Use the date format specified in the NLS_LANG and NLS_DATE_FORMAT environment variables. You can also use data constants such as SYSDATE to specify the time. (SYSDATE – 5 means 5 days earlier than the system date).

RMAN recovers tables or table partitions to the state they were at the time specified by the log sequence number and thread number. RMAN selects only files that it can use to restore or recover up to but not including the specified sequence number.

Process Steps of Table Recovery

1. Perform the planning tasks and start an RMAN session with the CONNECT TARGET command.
2. Enter the RECOVER TABLE command.
3. RMAN determines the backup based on your specification.
4. RMAN creates an auxiliary instance by using the AUXILIARY DESTINATION clause, if specified.
5. RMAN recovers your tables or table partitions, up to the specified point in time, into this auxiliary instance.
6. RMAN creates a Data Pump export dump file that contains the recovered objects with the DUMP FILE=*name* and DATAPUMP DESTINATION=<OS path>.

Note: If a file with the name specified by DUMP FILE exists in the location in which the dump file must be created, then the export fails.



Copyright © 2020, Oracle and/or its affiliates.

1. After verifying the prerequisites, start an RMAN session and connect to the target instance.
2. Enter the RECOVER TABLE command with your required clauses.
3. RMAN determines the backup that contains the data that needs to be recovered, based on the point in time specified for recovery.
4. RMAN creates an auxiliary instance. Optionally, you can specify the location of the auxiliary instance files with the AUXILIARY DESTINATION or SET NEWNAME clauses of the RECOVER command. AUXILIARY DESTINATION is the recommended clause, because if you use SET NEWNAME and you forget just one data file name, the recovery would not happen.
5. RMAN recovers the specified tables or table partitions, up to the specified point in time, into this auxiliary instance.
6. RMAN creates a Data Pump export dump file that contains the recovered objects. You can optionally specify the name of the export dump file (with the DUMP FILE clause, default OS-specific name) that is used to store the metadata from the source database. You can also specify the location in which the export dump file is created with the DATAPUMP DESTINATION clause. The location is typically the path of the OS directory that stores the dump file. If omitted, the dump file is stored in the AUXILIARY DESTINATION location. If that is not specified, then the dump file is stored in a default OS-specific location.

Process Steps of Table Recovery

7. RMAN imports the recovered objects into the target database unless you specified NOTABLEIMPORT.
8. RMAN optionally renames the recovered tables or table partitions with the REMAP TABLE and the REMAP TABLESPACE clauses. (Existing objects are not changed.)

Note: If you remap a table, the dependent objects are excluded from the import. You must re-create indexes and constraints.



Copyright © 2020, Oracle and/or its affiliates.

7. By default, RMAN imports the recovered objects that are stored in the export dump file into the target database. However, you can choose not to import the recovered objects by using the NOTABLEIMPORT clause of the RESTORE command.
If you choose not to import the recovered objects, RMAN recovers the tables or table partitions to the specified point and then creates the export dump file. However, this dump file is not imported into the target database. You must manually import this dump file into your target database, when required, by using the Data Pump Import utility.
8. RMAN optionally renames the recovered tables or table partitions in the target database with the REMAP TABLE option.
 - If a table already exists and the REMAP option is not specified, then the table recovery generates an error.
 - If the REMAP option is specified, then the indexes and constraints are not imported. You must create dependent objects yourself.

To import the recovered objects into a tablespace that is different from the one in which the objects originally existed, use the REMAP TABLESPACE clause of the RECOVER command. Only the tables or table partitions that are being recovered are remapped; the existing objects are not changed.

Summary

In this lesson, you should have learned how to:

- Distinguish and describe point-in-time recovery (PITR) of table, tablespace, and database
- Identify the circumstances where PITR is a good solution and where it cannot be used
- List what operations occur when you perform a PITR
- Determine the correct target time for the PITR
- Perform automated TSPITR
- Perform table recovery from backups



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Recovering from Media Failure: Incomplete Recovery
- Recovering a Table from a Backup



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Performing Block Media Recovery

The Oracle logo, consisting of the word "ORACLE" in white capital letters inside a red square.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Proactively check for block corruption
- Perform block media recovery



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

What Is Block Corruption?

- Whenever a block is read or written, a consistency check is performed.
 - Block version
 - DBA (data block address) value in cache as compared to the DBA value in the block buffer
 - Block-checksum, if enabled
- A corrupt block is identified as being one of the following:
 - Media corrupt
 - Logically (or software) corrupt



Copyright © 2020, Oracle and/or its affiliates.

A corrupted data block is a block that is not in a recognized Oracle format or whose contents are not internally consistent. Typically, corruptions are caused by faulty hardware or operating system problems. The Oracle database identifies corrupt blocks as either “logically corrupt” or “media corrupt.” If it is logically corrupt, there is an Oracle internal error. Logically corrupt blocks are marked corrupt by the Oracle database after it detects the inconsistency. If it is media corrupt, the block format is not correct; the information in the block does not make any sense after being read from disk.

Corrupt blocks are discovered when they are accessed or during a validation check. Validation consists of checking all blocks in the object that have not previously been marked corrupt. For each block, the transaction and data layer portions are checked for self-consistency. During `CHECK_OBJECT`, if a block is encountered that has a corrupt buffer cache header, then that block is skipped.

As you just learned, a number of data failures and corruptions can be repaired with the Data Recovery Advisor. Now you learn about a manual way to diagnose and repair corruptions.

You can repair a media corrupt block by recovering the block or dropping the database object that contains the corrupt block or both. If media corruption is due to faulty hardware, the problem will not be completely resolved until the hardware fault is corrected.

Block Corruption Symptoms: ORA-01578

The error `ORA-01578: "ORACLE data block corrupted (file # %s, block # %s)"`:

- Is generated when a corrupted data block is found
- Always returns the tablespace relative file number and block number
- Is returned to the session that issued the query being performed when the corruption was discovered
- Appears in the `alert.log` file



Copyright © 2020, Oracle and/or its affiliates.

Usually, the ORA-01578 error is the result of a hardware problem. If the ORA-01578 error is always returned with the same arguments, it is most likely a media corrupt block.

If the arguments change each time, there may be a hardware problem, and you should have the memory and page space checked and the I/O subsystem checked for bad controllers.

Note: ORA-01578 returns the tablespace relative file number, but the accompanying ORA-01110 error displays the absolute file number.

How to Handle Corruption

- Check the alert log and operating system log file.
- Use available diagnostic tools to find out the type of corruption.
- Determine whether the error persists by running checks multiple times.

```
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE ONLINE;
```
- Recover data from the corrupted object if necessary.
- Resolve any hardware issues:
 - Memory boards
 - Disk controllers
 - Disks
- Recover or restore data from the corrupt object if necessary.



Copyright © 2020, Oracle and/or its affiliates.

Always try to find out whether the error is permanent. Run the ANALYZE command multiple times or, if possible, perform a shutdown and a startup and try again to perform the operation that failed earlier. Find out whether there are more corruptions. If you encounter one, there may be other corrupted blocks as well.

Hardware failures should be addressed immediately. When you encounter hardware problems, the vendor should be contacted and the machine should be checked and fixed before continuing. A full hardware diagnostics session should be run.

Many types of hardware failures are possible:

- Faulty I/O hardware or firmware
- Operating system I/O or caching problem
- Memory or paging problems
- Disk repair utilities

For more information, see *Oracle Database Administrator's Guide*.

Setting Parameters to Detect Corruption

Parameter	Use
DB_BLOCK_CHECKING	Prevent memory and data corruption
DB_BLOCK_CHECKSUM	Detect I/O storage, disk corruption



Tip: Test first, because these parameters have a performance impact.

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

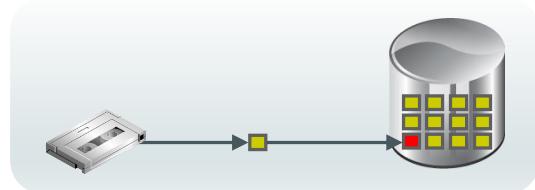
Recommended generic block-corruption parameters:

- DB_BLOCK_CHECKING, which initiates checking of database blocks. This check can often prevent memory and data corruption. (Default: FALSE, recommended: FULL or MEDIUM)
- DB_BLOCK_CHECKSUM, which initiates the calculation and storage of a checksum in the cache header of every data block when writing it to disk. Checksums assist in detecting corruption caused by underlying disks, storage systems, or I/O systems. (Default: TYPICAL, recommended: FULL)

Note: Each of the parameters set to detect corruption impacts performance to varying degrees and thus requires testing before production use.

Block Media Recovery

- Block media recovery:
 - Lowers the mean time to recover (MTTR)
 - Increases availability during media recovery
 - The data file remains online during recovery.
 - Only blocks being recovered are inaccessible.
 - Is invoked using the RMAN RECOVER...BLOCK command
 - Restores blocks by using flashback logs and full or level 0 backups
 - Media recovery is performed using redo logs.
- The `V$DATABASE_BLOCK_CORRUPTION` view displays blocks marked corrupt.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

In most cases, the database marks a block as media corrupt and then writes it to disk when the corruption is first encountered. No subsequent read of the block will be successful until the block is recovered. You can perform block recovery only on blocks that are marked corrupt or fail a corruption check. Block media recovery is performed using the RMAN RECOVER...BLOCK command. By default, RMAN searches the flashback logs for good copies of the blocks and then searches for the blocks in full or level 0 incremental backups. When RMAN finds good copies, it restores them and performs media recovery on the blocks. Block media recovery can use only redo logs for media recovery, not incremental backups.

The `V$DATABASE_BLOCK_CORRUPTION` view displays blocks marked corrupt by database components such as RMAN commands, ANALYZE, dbv, SQL queries, and so on. The following types of corruption result in rows added to this view:

- **Physical/Media corruption:** The database does not recognize the block: the checksum is invalid, the block contains all zeros, or the block header is fractured. Physical corruption checking is enabled by default.
- **Logical corruption:** The block has a valid checksum, the header and footer match, but the contents are inconsistent. Block media recovery cannot repair logical block corruption. Logical corruption checking is disabled by default. You can turn it on by specifying the `CHECK LOGICAL` option of the BACKUP, RESTORE, RECOVER, and VALIDATE commands.

Prerequisites for Block Media Recovery

- The target database must be in ARCHIVELOG mode.
- The backups of the data files containing the corrupt blocks must be full or level 0 backups.
 - Proxy copies must be restored to a nondefault location before they can be used.
- RMAN can use only archived redo logs for the recovery.
- The corrupted data block can be restored from Flashback Logs if available.



Copyright © 2020, Oracle and/or its affiliates.

The following prerequisites apply to the RECOVER . . . BLOCK command:

- The target database must run in ARCHIVELOG mode and be open or mounted with a current control file.
- The backups of the data files containing the corrupt blocks must be full or level 0 backups and not proxy copies. If only proxy copy backups exist, then you can restore them to a nondefault location on disk, in which case RMAN considers them data file copies and searches them for blocks during block media recovery.
- RMAN can use only archived redo logs for the recovery. RMAN cannot use level 1 incremental backups. Block media recovery cannot survive a missing or inaccessible archived redo log, although it can sometimes survive missing redo records.
- Flashback Database must be enabled on the target database for RMAN to search the flashback logs for good copies of corrupt blocks. If flashback logging is enabled and contains older, uncorrupted versions of the corrupt blocks, then RMAN can use these blocks, possibly speeding up the recovery.

Recovering Individual Blocks

The RMAN **RECOVER...BLOCK** command:

- Identifies the backups containing the blocks to recover
- Reads the backups and accumulates requested blocks into in-memory buffers
- Manages the block media recovery session by reading the archive logs from backup if necessary

```
RECOVER DATAFILE 6 BLOCK 3; Recover a single block

RECOVER          Recover multiple blocks
DATAFILE 2 BLOCK 43 in multiple data files
DATAFILE 2 BLOCK 79
DATAFILE 6 BLOCK 183;

RECOVER CORRUPTION LIST; Recover all blocks logged in
                         V$DATABASE_BLOCK_CORRUPTION
```

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Before block recovery can take place, you must identify the corrupt blocks. Typically, block corruption is reported in the following locations:

- Results of the **LIST FAILURE**, **VALIDATE**, or **BACKUP ... VALIDATE** command
- The **V\$DATABASE_BLOCK_CORRUPTION** view
- Error messages in standard output
- The alert log and user trace files (identified in the **V\$DIAG_INFO** view)
- Results of the **SQL ANALYZE TABLE** and **ANALYZE INDEX** commands
- Results of the **DBVERIFY** utility

For example, you may discover the following messages in a user trace file:

```
ORA-01578: ORACLE data block corrupted (file # 7, block # 3)
ORA-01110: data file 7: '/oracle/oradata/orcl/tools01.dbf'
ORA-01578: ORACLE data block corrupted (file # 2, block # 235)
ORA-01110: data file 2: '/oracle/oradata/orcl/undotbs01.dbf'
```

After the blocks have been identified, run the **RECOVER ... BLOCK** command at the RMAN prompt, specifying the file and block numbers for the corrupted blocks.

```
RECOVER
DATAFILE 7 BLOCK 3
DATAFILE 2 BLOCK 235;
```

Best Practice: Proactive Checks

Invoking proactive health check of the database and its components:

- Health Monitor or RMAN VALIDATE DATABASE command
- Checking for logical and physical corruption
- Findings logged in the ADR



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For very important databases, you may want to execute additional proactive checks (possibly daily during low peak interval periods). You can schedule periodic health checks through the Health Monitor or by using the RMAN VALIDATE command. In general, when a reactive check detects failure(s) in a database component, you may want to execute a more complete check of the affected component.

The RMAN VALIDATE DATABASE command is used to invoke health checks for the database and its components. It extends the existing VALIDATE BACKUPSET command. Any problem detected during validation is displayed to you. Problems initiate the execution of a failure assessment. If a failure is detected, it is logged into ADR as a finding. You can use the LIST FAILURE command to view all failures recorded in the repository.

The VALIDATE command supports validation of individual backup sets and data blocks. In a physical corruption, the database does not recognize the block at all. In a logical corruption, the contents of the block are logically inconsistent. By default, the VALIDATE command checks for physical corruption only. You can specify CHECK LOGICAL to check for logical corruption as well.

Block corruptions can be divided into interblock corruption and intrablock corruption. In intrablock corruption, the corruption occurs within the block itself and can be either physical or logical corruption. In interblock corruption, the corruption occurs between blocks and can be only logical corruption. The VALIDATE command checks for intrablock corruptions only.

Checking for Block Corruption

- The `VALIDATE` command:
 - Scans the specified files and verifies their contents
 - Confirms that the data files exist and are in the correct location
 - Checks for corrupt data blocks
 - Can check individual backup sets and data blocks
 - Skips never-used blocks
 - Displays failures and logs them in the ADR
- Specify the `CHECK LOGICAL` option to check for both physical and logical corruption.
- Query the `V$DATABASE_BLOCK_CORRUPTION` view to review output.



Copyright © 2020, Oracle and/or its affiliates.

The `VALIDATE` commands can be used to manually check for physical and logical corruptions in database files. You can also use the `BACKUP VALIDATE` command; however, only the `VALIDATE` command can be used to check individual backup sets and data blocks.

Refer to *Oracle Database Backup and Recovery Reference* for syntax detail.

Summary

In this lesson, you should have learned how to:

- Proactively check for block corruption
- Perform block media recovery



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Repairing Block Corruption



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Performing Additional Recovery Operations

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Recover from the loss of the server parameter file
- Recover from control file and redo log file failures
- Re-create the password authentication file



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Recovery from Loss of Server Parameter File

The FROM MEMORY clause allows the creation of current systemwide parameter settings.

```
SQL> CREATE PFILE [= 'pfile_name' ]
      FROM { { SPFILE [= 'spfile_name'] } | MEMORY } ;
```

```
SQL> CREATE SPFILE [= 'spfile_name' ]
      FROM { { PFILE [= 'pfile_name'] } | MEMORY } ;
```



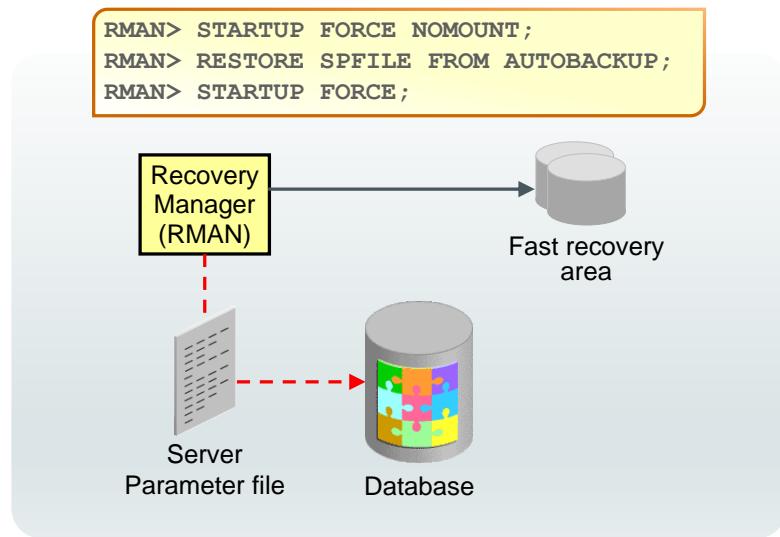
Copyright © 2020, Oracle and/or its affiliates.

The easiest way to recover a server parameter file is to use the FROM MEMORY clause, which creates a text initialization parameter file (PFILE) or server parameter file (SPFILE) by using the current system wide parameter settings. In a RAC environment, the created file contains the parameter settings from each instance.

During instance startup, all parameter settings are logged in to the alert.log file. The alert.log parameter dump text is written in valid parameter syntax. This facilitates cutting and pasting of parameters into a separate file and then using it as a PFILE for a subsequent instance. The name of the PFILE or SPFILE is written to alert.log at instance startup time. In cases when an unknown client-side PFILE is used, the alert log indicates this as well.

Restoring the Server Parameter File from the Control File Autobackup

```
RMAN> STARTUP FORCE NOMOUNT;
RMAN> RESTORE SPFILE FROM AUTOBACKUP;
RMAN> STARTUP FORCE;
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

If you have lost the server parameter file and you cannot use the `FROM MEMORY` clause, then you can restore it from the autobackup. The procedure is similar to restoring the control file from autobackup. If the autobackup is not in the flash recovery area, set the DBID for your database. Issue the `RESTORE SPFILE FROM AUTOBACKUP` command.

If you are restoring the `SPFILE` to a nondefault location, specify the command as follows:

```
RESTORE SPFILE TO <file_name> FROM AUTOBACKUP
```

If you are restoring the server parameter file from the fast recovery area, specify the command as follows:

```
RMAN> run {
2> restore spfile from autobackup
3> recovery area = '<flash recovery area destination>'
4> db_name = '<db_name>';
5> }
```

Loss of a Control File

If a control file is lost or corrupted, the instance normally aborts.

- If control files are stored in ASM disk groups, recovery options are as follows:
 - Perform guided recovery using Cloud Control.
 - Put the database in `NOMOUNT` mode and use an RMAN command to restore the control file from existing control file.
- If control files are stored as regular file system files, then:
 - Shut down the database
 - Copy the existing control file to replace the lost control file

```
RMAN> restore controlfile from
      '+DATA/orcl/controlfile/current.260.695209463' ;
```

Open the database after the control file is successfully restored.



Copyright © 2020, Oracle and/or its affiliates.

The options for recovery from the loss of a control file depend on the storage configuration of the control files and on whether at least one control file remains or all have been lost.

If you are using ASM storage, and at least one control file copy remains, you can perform guided recovery by using Enterprise Manager Cloud Control or perform manual recovery by using RMAN as follows:

1. Put the database in `NOMOUNT` mode.
2. Connect to RMAN and issue the `RESTORE CONTROLFILE` command to restore the control file from an existing control file, for example:
`restore controlfile from
 '+DATA/orcl/controlfile/current.260.695209463' ;`
3. After the control file is successfully restored, open the database.

Note: You can also use the `ASMCMD cp` command to restore the control file.

If your control files are stored as regular file system files and at least one control file copy remains, then, while the database is down, you can just copy one of the remaining control files to the missing file's location. If the media failure is due to the loss of a disk drive or controller, copy one of the remaining control files to some other location and update the instance's parameter file to point to the new location. Alternatively, you can delete the reference to the missing control file from the initialization parameter file. Remember that Oracle recommends having at least two control files at all times.

Recovering from the Loss of All Control File Copies: Overview

	Current	Backup
Available	Restore backup control file, perform complete recovery, OPEN RESETLOGS.	Restore backup control file, perform complete recovery, OPEN RESETLOGS.
Unavailable	Re-create control file, OPEN RESETLOGS.	Restore backup control file, perform point-in-time recovery, OPEN RESETLOGS.

```

graph TD
    A[Online log status] --> Available[Available]
    B[Data file status] --> Unavailable[Unavailable]
  
```



Copyright © 2020, Oracle and/or its affiliates.

Loss of all control files should never happen. **Prevention is better than recovery.** Even though you have copies of the control file stored in different locations, there is still the possibility that you will, at some point, have to recover from losing all those copies. If you have lost all copies of the current control file, and have a backup control file, your course of action depends on the status of the online log files and the data files. If you do not have a backup of the control file but have a text file created with the `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command, you may be able to use it to re-create the control file.

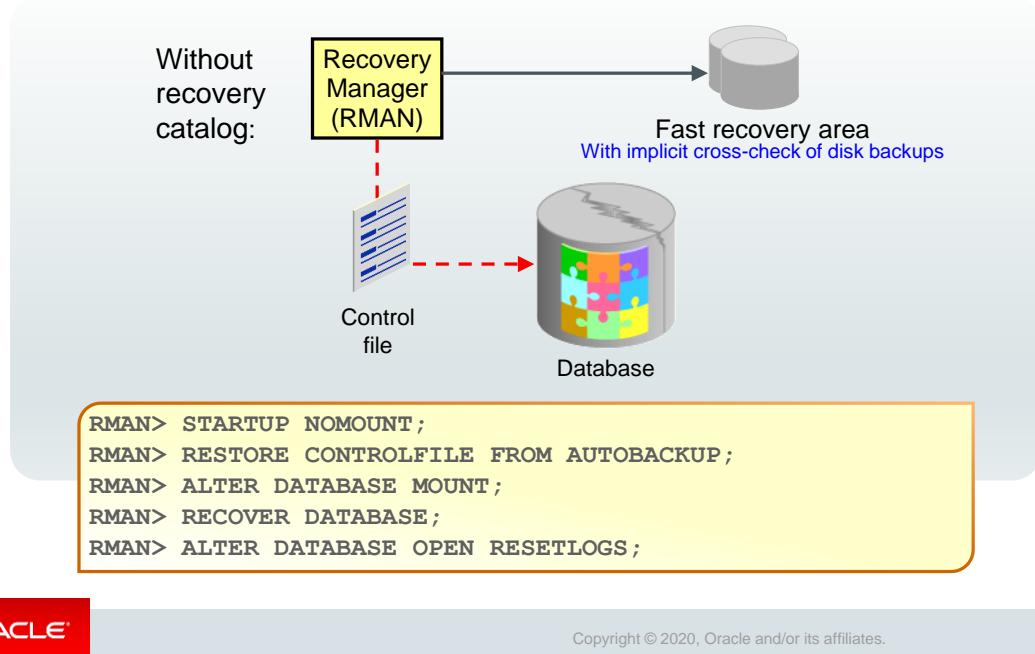
Online Logs Available

If the online logs are available and contain redo necessary for recovery, and the data files are current, then you can restore a backup control file, perform complete recovery, and open the database with the `RESETLOGS` option. You must specify the file names of the online redo logs during recovery. If the data files are not current, perform the same procedure.

Online Logs Not Available

If the online logs are not available, and the data files are current, then re-create the control file and open `RESETLOGS`. However, if the data files are not current, restore a backup control file, perform point-in-time recovery, and open `RESETLOGS`.

Restoring the Control File from Autobackup



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Oracle Corporation recommends that you should have AUTOBACKUP of the control file configured so that you are able to quickly restore the control file if needed. The commands used for restoring your control file are the same, whether or not you are using a fast recovery area. However, if you are using a fast recovery area, RMAN implicitly cross-checks backups and image copies listed in the control file and catalogs any files in the fast recovery area that are not recorded in the restored control file, thereby improving the usefulness of the restored control file in the restoration of the rest of your database.

Use the commands shown in the slide to recover from lost control files:

1. First, start the instance in `NOMOUNT` mode. It cannot be mounted because there is no control file.
2. Restore the control file from backup.
3. Now that there is a control file, you can mount the database.
4. You must recover the database, because you now have a backup control file that contains information about an older version of the database.
5. After recovering the database, you can open it. You must specify `RESETLOGS` because the new control file represents a different instantiation of the database.

Note: Tape backups are not automatically cross-checked after the restoration of a control file. After restoring the control file and mounting the database, you must cross-check the backups on tape.

Restoring the SPFILE and the Control File

With recovery catalog:

- Database in NOMOUNT state

```
RMAN> RESTORE CONTROLFILE;  
RMAN> RESTORE CONTROLFILE... TO <destination>
```

Loss of SPFILE and control file:

1. Set the DBID or use recovery catalog.
2. Restore the SPFILE from the autobackup.
3. Start the instance with the restored SPFILE.
4. Restore the control file from the autobackup.
5. Mount the database with the restored control file.
6. Restore and recover the database.
7. Open the database with the RESETLOGS option.

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

If you have a recovery catalog, you do not have to set the DBID or use the control file autobackup to restore the control file. You can use the RESTORE CONTROLFILE command with no arguments:

```
RMAN> RESTORE CONTROLFILE;
```

The instance must be in the NOMOUNT state when you perform this operation, and RMAN must be connected to the recovery catalog. The restored control file is written to all locations listed in the CONTROL_FILES initialization parameter.

Use the RESTORE CONTROLFILE... TO <destination> command to restore the control file to a non-default location.

If you have also lost the SPFILE for the database and need to restore it from the autobackup, the procedure is similar to restoring the control file from autobackup. You must first set the DBID for your database and then use the RESTORE SPFILE FROM AUTOBACKUP command.

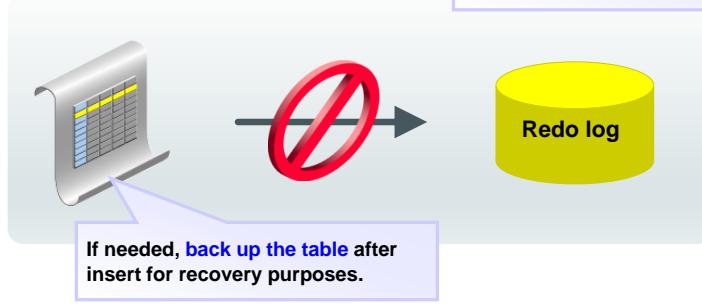
After you have started the instance with the restored server parameter file, RMAN can restore the control file from the autobackup. After you restore and mount the control file, you have the backup information necessary to restore and recover the database.

After restoring the control files of your database from backup, you must perform complete media recovery and then open your database with the RESETLOGS option.

Recovering NOLOGGING Database Objects

```
SQL> CREATE TABLE sales_copy NOLOGGING;
SQL> INSERT /*+ APPEND */ INTO sales_copy
2  SELECT * FROM sales_history;
```

If an object is created with the NOLOGGING clause, direct-path inserts cannot be recovered.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Take advantage of the efficiencies of the NOLOGGING attribute of tables and indexes if you can. When you create a table as NOLOGGING, minimal redo data is written to the redo stream to support the creation of the object. This is useful for making large inserts go faster.

In the example in the slide, the SALES_COPY table is created as a NOLOGGING table. As a result, when an insert is done with the APPEND hint, no redo is generated for that particular insert statement. As a result, **you cannot recover this transaction** on the SALES_HISTORY table. If that is a problem, it is important that you make a backup of whatever tables you populate in this way, right afterward. Then you are able to go to the more recent backup of the table.

If you perform media recovery, and there are NOLOGGING objects involved, they will be marked logically corrupt during the recovery process. In this case, drop the NOLOGGING objects and re-create them.

Loss of a Redo Log File

If a member of a redo log file group is lost and if the group still has at least one member, note the following results:

- Normal operation of the instance is not affected.
- You receive a message in the alert log notifying you that a member cannot be found.
- You can restore the missing log file by dropping the lost redo log member and adding a new member.
- If the group with the missing log file has been archived, you can clear the log group to re-create the missing file.
- Immediately take a full database backup of the whole database.



Copyright © 2020, Oracle and/or its affiliates.

Recovering from the loss of a single redo log group member should not affect the instance.

To perform this recovery:

1. Determine whether there is a missing log file by examining the alert log.
2. Restore the missing file by first dropping the lost redo log member:

```
ALTER DATABASE DROP LOGFILE MEMBER ''
```

Then add a new member to replace the lost redo log member:

```
ALTER DATABASE ADD LOGFILE MEMBER '' TO GROUP n
```

Cloud Control can also be used to drop and re-create the log file member.

Note: If using Oracle Managed Files (OMF) for your redo log files and you use the preceding syntax to add a new redo log member to an existing group, that new redo log member file will not be an OMF file. If you want to ensure that the new redo log member is an OMF file, then the easiest recovery option would be to create a new redo log group and then drop the redo log group that had the missing redo log member.

3. If the media failure is due to the loss of a disk drive or controller, rename the missing file.
4. If the group has already been archived, or if you are in NOARCHIVELOG mode, you may choose to solve the problem by clearing the log group to re-create the missing file or files. Select the appropriate group and then select the Clear Logfile action. You can also clear the affected group manually with the following command:

```
ALTER DATABASE CLEAR LOGFILE GROUP #
```

Note: Enterprise Manager Cloud Control does not allow you to clear a log group that has not been archived. Doing so breaks the chain of redo information. If you must clear an unarchived log group, you should *immediately* take a full backup of the whole database. Failure to do so may result in a loss of data if another failure occurs. To clear an unarchived log group, use the following command:
`ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP #`

Log Group Status: Review



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

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

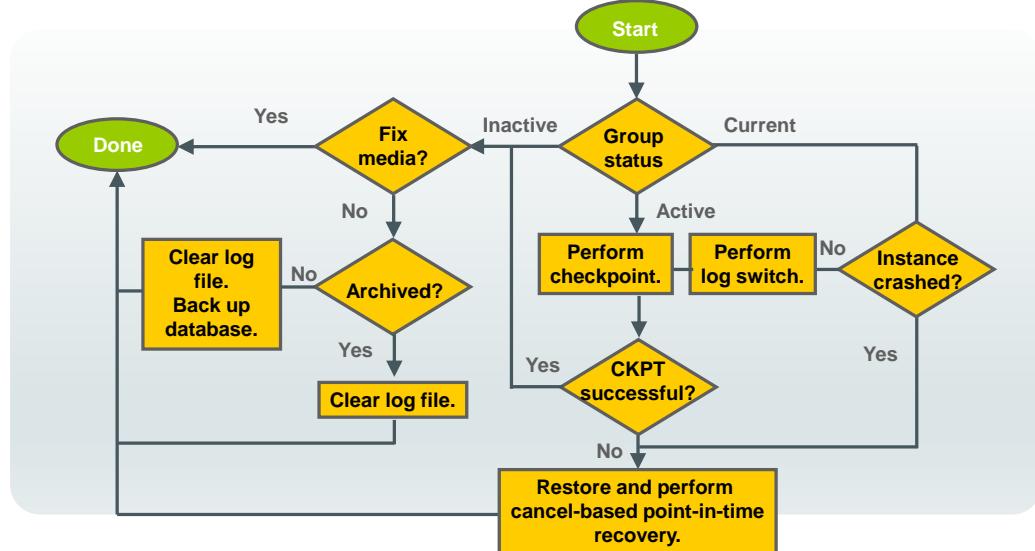
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

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

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

Recovering from the Loss of a Redo Log Group



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

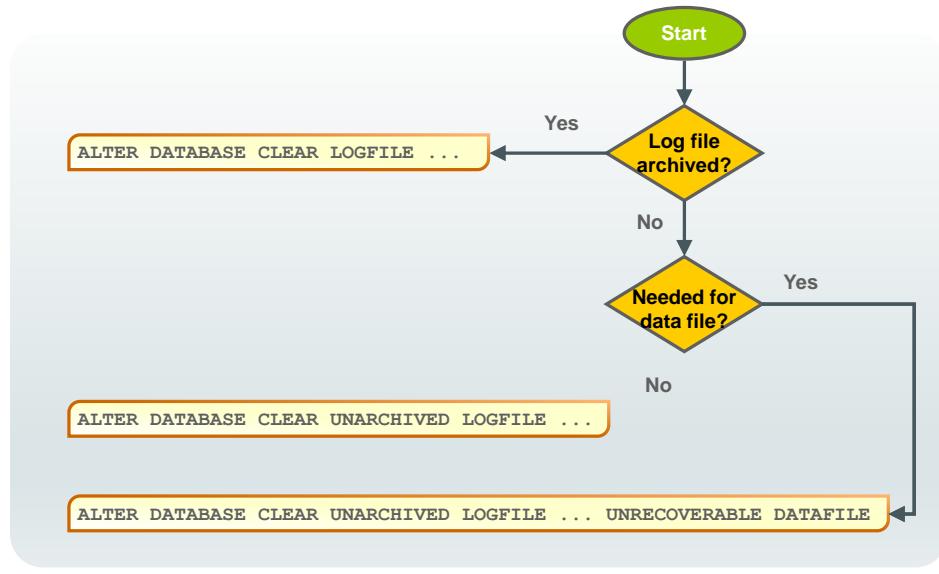
If you have lost an entire redo log group, then all copies of the log files for that group are unusable or gone.

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

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

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

Clearing a Log File



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Clear a log file using this command:

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

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

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

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

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

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

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

Re-creating a Password Authentication File

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

To recover from the loss of a password file:

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

```
$ orapwd file=$ORACLE_HOME/dbs/orapworcl password=ora entries=20
```

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



Copyright © 2020, Oracle and/or its affiliates.

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

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

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

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

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

Using a Password File

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

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

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

where:

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

Example: `orapwd file=$ORACLE_HOME/dbs/orapwU15 password=admin entries=20`

Other options (in release 12.1 and later) include:

- **ASM**: Set to **y**, creates an Oracle ASM password file. The default **n** creates a database password file.
- **FORMAT**: Set to **12** (the default), the password file is created in the release 12c format. This format supports the **SYSBACKUP**, **SYSDG**, and **SYSKM** system privileges. If set to **legacy**, the password file is in the legacy format, which is the format before Oracle Database 12c. This argument cannot be set to **legacy** when the **SYSBACKUP** or the **SYSDG** argument is specified.
- **SYSBACKUP**: **y** creates a **SYSBACKUP** entry in the password file. You are prompted for the password. The password is stored in the created password file.
- **SYSDG**: **y** creates a **SYSDG** entry in the password file. You are prompted for the password. The password is stored in the created password file.
- **SYSKM**: **y** creates a **SYSKM** entry in the password file. You are prompted for the password. The password is stored in the created password file.
- **DELETE**: **y** deletes the specified password file. If set to the default **n**, then the specified password file is created.
- **FORCE**: **y** permits overwriting an existing password file.

For a complete list of options, see *Oracle Database Administrator’s Guide*.

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

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

Password File Locations

UNIX: `$ORACLE_HOME/dbs`

Windows: `%ORACLE_HOME%\database`

Maintaining the Password File

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

Summary

In this lesson, you should have learned how to:

- Recover from the loss of the server parameter file
- Recover from control file and redo log file failures
- Re-create the password authentication file



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Recovering from the Loss of a Parameter File
- Restoring the Control File
- Recovering from the Loss of All Control Files
- Restoring the Password File



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Oracle Flashback Technology: Overview

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to explain how to use flashback technologies to protect against and recover from various types of errors.

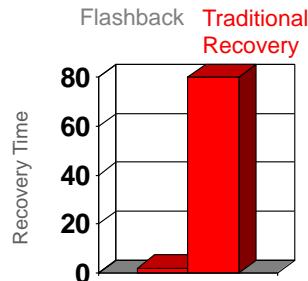


ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Flashback Technologies Error Detection and Correction

- Flashback makes error recovery much easier by:
 - Enabling you to view data as of a past point in time
 - “Rewinding” unwanted data changes
 - Minimizing the time it takes to correct an error
- Flashback is easy to use and includes simple commands, with no complex procedures.

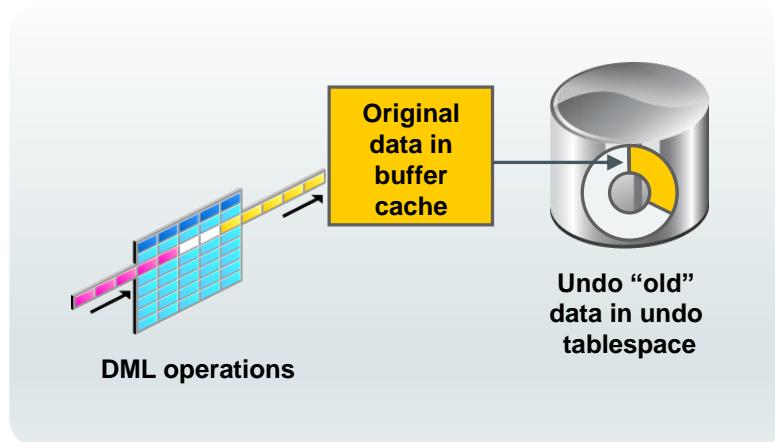


ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Oracle Database Flashback technologies are a set of data recovery solutions that provide capabilities to correct human errors by selectively and efficiently undoing the effects of a mistake. Flashback technologies support recovery at all levels including row, transaction, table, and the entire database.

Review: Transactions and Undo



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

When a transaction starts, it is assigned to an undo segment. Throughout the life of the transaction, when data is changed, the original “old” values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the V\$TRANSACTION view.

Undo segments are specialized segments that are automatically created by the instance as needed to support transactions. Like all segments, undo segments are made up of extents, which, in turn, consist of undo blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions.

When transactions fill the blocks in their current undo segment extent, they are assigned another block in the same extent. If no free blocks remain in that extent, the transaction acquires a block from the next extent in the segment. If all extents are in use, the transaction either wraps around back into the first extent or requests that a new extent be allocated to the undo segment if by wrapping it would enter an extent containing active undo.

The diagram in the slide shows a table icon on the left with original data arriving from a DML operation. The original data is kept in the buffer cache (if not aged out) and then written to the undo tablespace (shown in circular form on the right).

Flashback Technology



Using Flashback technology:

- Viewing past states of data
- Winding data back and forth in time
- Assisting users in error analysis and recovery

For error analysis:

Oracle Flashback Query

Oracle Flashback Version Query

Oracle Flashback Transaction Query

For error recovery:

Oracle Flashback Transaction Backout

Oracle Flashback Table

Oracle Flashback Drop

Oracle Flashback Database

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Oracle Flashback technology is a group of features that support viewing past states of data—and winding data back and forth in time—without requiring restoring the database from backup. With this technology, you help users analyze and recover from errors.

- **Flashback Query:** View committed data as it existed at some point in the past. The SELECT command with the AS OF clause references a time in the past through a time stamp or system change number (SCN).
- **Flashback Version Query:** View committed historical data for a specific time interval. Use the VERSIONS BETWEEN clause of the SELECT command (for performance reasons with existing indexes).
- **Flashback Transaction Query:** View changes made at the transaction level.
- **Flashback Transaction Backout:** Roll back a specific transaction and dependent transactions.
- **Flashback Table:** Rewind one or more tables to their contents at a previous time without affecting other database objects.
- **Flashback Drop:** Reverse the effects of dropping a table by returning the dropped table from the recycle bin to the database along with dependent objects such as indexes and triggers.
- **Flashback Database:** Return the database to a past time or SCN.

Flashback Technology

Object Level	Scenario Examples	Flashback Technology	Depends On	Affects Data
Database	Truncate table, undesired multitable changes made	Database	Flashback logs	TRUE
Table	Drop table.	Drop	Recycle bin	TRUE
	Update with the wrong WHERE clause.	Table	Undo data	TRUE
	Compare current data with data from the past.	Query	Undo data	FALSE
Transaction	Compare versions of a row.	Version Query	Undo data	FALSE
	Investigate and back out suspect transactions.	Transaction Query	Undo/redo from archive logs	TRUE
Table and transaction	Audit, compliance, historical reports, ILM	Data Archive (Temporal)	Tablespace	FALSE



Copyright © 2020, Oracle and/or its affiliates.

You can use Flashback technology when a logical corruption occurs in the Oracle database and you need to recover data quickly and easily. As with human errors, it is difficult to identify the objects and rows that are affected by an erroneous transaction. With Flashback technology, you can diagnose how errors were introduced into the database and then repair the damage. You can view the transactions that have contributed to specific row modifications, view the entire set of versions of a given row during a specific time period, or just view data as it appeared at a specific time in the past. The table in the slide shows typical uses of Flashback technology. Flashback Database depends on the flashback logs to perform the Flashback Database operation. Flashback Drop uses the recycle bin. All other techniques use undo data.

Not all flashback features modify the database. Some are simply methods to query other versions of data. You can use these features to investigate a problem and aid in recovery:

- Determine the type of flashback operation to use to correct the problem.
- Use the result of the query in an `INSERT`, `UPDATE`, or `DELETE` that enables you to repair the erroneous data.

Summary

In this lesson, you should have learned how to use flashback technologies to protect against and recover from various types of errors.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Using Logical Flashback Features

The Oracle logo, consisting of the word "ORACLE" in white capital letters inside a red square.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Explain how to use flashback technologies to protect against and recover from various types of errors
- Perform flashback operations
- Distinguish between temporal validity and temporal history



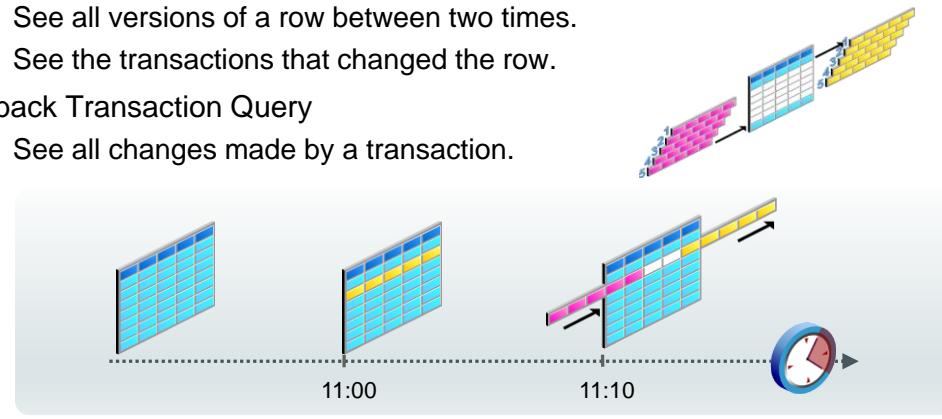
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Using Flashback Technology to Query Data

Flashback
➤ - Query
- Versions
- Table
- Transaction
- Drop
- Data Archive

- Flashback Query
 - Query all data at a specified point in time.
- Flashback Version Query
 - See all versions of a row between two times.
 - See the transactions that changed the row.
- Flashback Transaction Query
 - See all changes made by a transaction.



ORACLE®

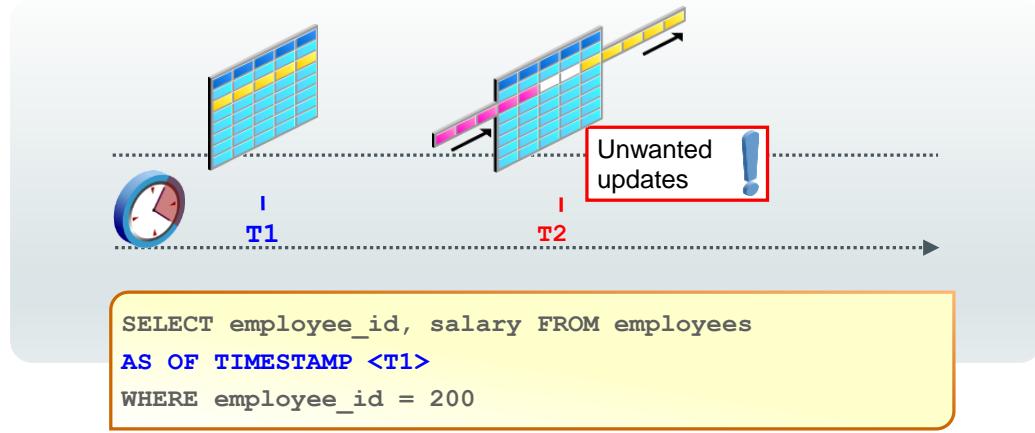
Copyright © 2020, Oracle and/or its affiliates.

Flashback technology provides the capability to query past versions of schema objects, query historical data, and perform change analysis. Every transaction logically generates a new version of the database. With Flashback technology, you can navigate through these versions to find an error and its cause:

- **Flashback Query:** Query all data as it existed at a specific point in time.
- **Flashback Version Query:** See all versions of rows between two times and the transactions that changed the row.
- **Flashback Transaction Query:** See all changes made by a transaction and, if needed, roll back a transaction with “undo” SQL commands.

Flashback Query

Use to query all data at a specified point in time or SCN.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

With the Flashback Query feature, you can perform queries as of a certain time. By using the AS OF clause of the SELECT statement, you can specify the time stamp for which to view the data. This is useful for analyzing a data discrepancy.

Note: TIMESTAMP and SCN are valid options for the AS OF clause.

See *Oracle Database SQL Language Reference* for detail on using the following functions:

- SCN_TO_TIMESTAMP: Returns the approximate timestamp associated with a specified SCN
- TIMESTAMP_TO_SCN: Returns the approximate system change number (SCN) associated with a specified timestamp

Flashback Version Query

Flashback
- Query
➤ - Versions
- Table
- Transaction
- Drop
- Data Archive

The VERSIONS clause:

- Retrieves all the versions of the rows that exist between two points in time or two SCNs
- Retrieves only committed data
- *Cannot* be used to query external tables, temporary tables, fixed tables, or views
- Can be used to create views
- Cannot span DDL commands
- Filters out segment shrink operations

```
SELECT versions_xid, salary FROM employees
VERSIONS BETWEEN TIMESTAMP <t1> and <t2>
WHERE employee_id = 200;
```

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

The Flashback Version Query feature enables you to use the VERSIONS clause to retrieve all the versions of the rows that exist between two points in time or two SCNs. The rows returned represent a history of changes for the rows across transactions.

Flashback Version Query retrieves only committed data. Uncommitted row versions within a transaction are not shown. The rows returned also include deleted and subsequently reinserted versions of the rows.

Use row history to audit the rows of a table and retrieve information about the transactions that affected the rows. You can then use the returned transaction identifier (the VERSIONS_XID pseudocolumn) to perform transaction mining by using LogMiner or to perform a Flashback Transaction Query.

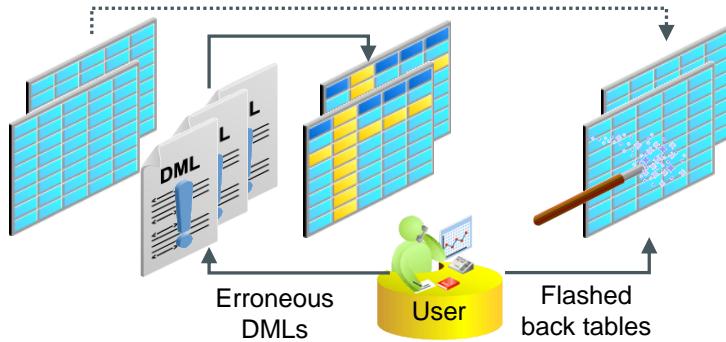
- The VERSIONS clause *cannot* be used to query external tables, temporary tables, fixed tables, or views. However, you can create a view with the VERSIONS clause.
- The VERSIONS clause in a SELECT statement cannot produce versions of rows across the DDL statements that change the structure of the corresponding tables. This means that the query stops producing rows after it reaches a time in the past when the table structure was changed.
- Certain maintenance operations, such as a segment shrink, may move table rows across blocks. In this case, the version query filters out such phantom versions because the row data remains the same.

Flashback Table: Overview

- Flashback Table recovers tables to a specific point in time.
- Flashback Table is an in-place operation.
- The database stays online.

Flashback

- Query
- Versions
- - Table
- Transaction
- Drop
- Data Archive



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

With Flashback Table, you can recover a set of tables to a specific point in time without having to perform traditional point-in-time recovery operations.

A Flashback Table operation is done in place, while the database is online, by rolling back only the changes that are made to the given tables and their dependent objects.

A Flashback Table statement is executed as a single transaction. All tables must be flashed back successfully, or the entire transaction is rolled back.

Note: You can use Flashback Version Query and Flashback Transaction Query to determine the appropriate flashback time.

Flashback Table

- Recovering a table or tables with associated objects to a specific point in time without restoring a backup
- Using data from the undo tablespace
- Prerequisites:
 - The FLASHBACK ANY TABLE or the FLASHBACK object privilege on the specific table
 - SELECT, INSERT, DELETE, and ALTER privileges on the table
 - Enabled row movement
- Interfaces: Cloud Control and SQL*Plus

```
FLASHBACK TABLE hr.departments  
TO_TIMESTAMP('2019-01-25 21:00:00', 'YYYY-MM-DD HH24:MI:SS');
```



Copyright © 2020, Oracle and/or its affiliates.

Using Flashback Table, you can recover a table or tables to a specific point in time without restoring a backup. When you use this feature, the data in tables and their associated objects (indexes, constraints, triggers, and so on) is restored. The data used to satisfy a Flashback Table request is retrieved from the undo tablespace. You can use Flashback Version Query and Flashback Transaction Query to determine the appropriate flashback time.

Flashback Table provides a way for users to easily and quickly recover from accidental modifications without a database administrator's involvement. You must grant the FLASHBACK TABLE or FLASHBACK ANY TABLE system privilege to any user that uses the Flashback Table feature. In addition, you must grant the SELECT, INSERT, DELETE, and ALTER object privileges to the user.

You can use Enterprise Manager or SQL*Plus to flash back a table. The wizard guides you through the process.

You must enable row movement on a table to be able to flash back the table. When you enable row movement, the Oracle server can move a row in the table. You can enable row movement by using Enterprise Manager or by using the ALTER TABLE command.

Flashback Table: Considerations

- The `FLASHBACK TABLE` command executes as a single transaction, acquiring exclusive DML locks.
- Statistics are not flashed back.
- Current indexes and dependent objects are maintained.
- Flashback Table operations:
 - Cannot be performed on system tables
 - Cannot span DDL operations
 - Generate undo and redo data



Copyright © 2020, Oracle and/or its affiliates.

The entire `FLASHBACK TABLE` statement is executed within a single transaction. All or none of the specified tables are flashed back.

Flashback Table acquires exclusive data manipulation language (DML) locks on all tables that are specified in the statement over the period of time the operation is in progress.

Statistics of impacted objects are not flashed back.

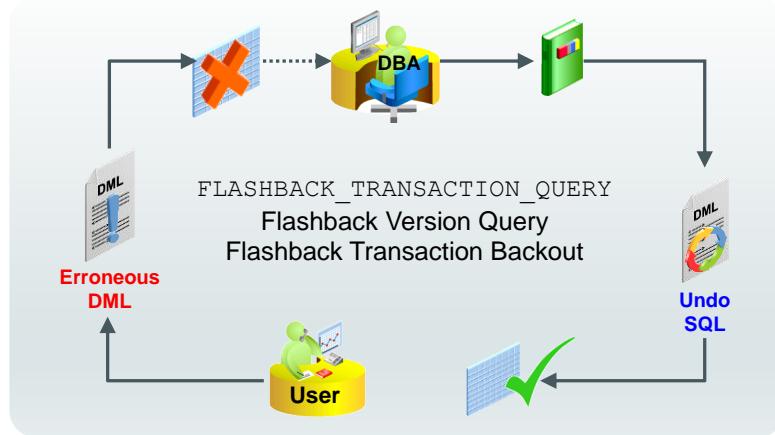
All existing indexes are maintained. Dropped indexes are not re-created. Dependent on-commit materialized views are also maintained automatically.

Tables specified in the `FLASHBACK TABLE` statement are flashed back, provided that none of the table constraints are violated. If any constraints are violated during flashback execution, the operation is aborted and the tables are left in the same state as they were just before the `FLASHBACK TABLE` statement invocation.

You cannot perform Flashback Table to a particular time that is older than the time of the execution of a data definition language (DDL) operation that altered the structure of or shrunk a table that would be involved in the flashback operation. This restriction does not apply to DDL statements that only change storage attributes of the tables.

Flashback Table cannot be performed on system tables, remote tables, and fixed tables.

Flashback Transaction Query



Flashback
- Query
- Versions
- Table
➤ - Transaction
- Drop
- Data Archive

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Flashback Transaction Query is a diagnostic tool that you can use to view changes made to the database at the transaction level. This enables you to diagnose problems in your database and perform analysis and audits of transactions (as shown in the graphic).

You can use the `FLASHBACK_TRANSACTION_QUERY` view to determine all the necessary SQL statements that can be used to undo the changes made either by a specific transaction or during a specific period of time.

This feature is used in conjunction with the Flashback Version Query feature with the help of the Perform Recovery Wizard.

Flashback Transaction Query: Considerations

- DDL commands are seen as dictionary updates.
- Flashback Transaction Query on a transaction underlying a DDL command displays the data dictionary changes.
- Dropped objects appear as object numbers.
- Dropped users appear as user identifiers.



Copyright © 2020, Oracle and/or its affiliates.

Within the database, DDL operations are nothing but a series of space management operations and changes to the data dictionary. Flashback Transaction Query on a transaction underlying a DDL command displays the changes made to the data dictionary.

When Flashback Transaction Query involves tables that have been dropped from the database, the table names are not reflected. Instead, object numbers are used.

If the user who executed a transaction is dropped, Flashback Transaction Query of that transaction displays the corresponding user ID only, and not the username.

Note: When there is not enough undo data for a specific transaction, a row with a value of UNKNOWN in the OPERATION column of FLASHBACK_TRANSACTION_QUERY is returned.

Flashback Transaction Backout

- Use Flashback Transaction to reverse a transaction and dependent transactions.
- Oracle Database determines the dependencies between transactions and, in effect, creates a compensating transaction that reverses the unwanted changes.
- Supplemental logging must be enabled.
- SELECT, FLASHBACK, and DML privileges on all affected tables must be granted.



Copyright © 2020, Oracle and/or its affiliates.

With Flashback Transaction, you can reverse a transaction and dependent transactions. Oracle Database determines the dependencies between transactions and, in effect, creates a compensating transaction that reverses the unwanted changes. The database rewinds to a state as if the transaction, and any transactions that could be dependent on it, never occurred.

You can use the Flashback Transaction functionality from within Enterprise Manager or by using PL/SQL packages.

To flash back or back out a transaction—that is, to create a compensating transaction—you must have the SELECT, FLASHBACK, and DML privileges on all affected tables.

Flashing Back a Transaction

- Use Enterprise Manager or the `DBMS_FLASHBACK.TRANSACTION_BACKOUT` procedure.
- If the PL/SQL call finishes successfully, it means that the transaction does not have any dependencies and a single transaction is backed out successfully.
- After choosing your back-out option, the dependency report is generated in the `DBA_FLASHBACK_TXN_STATE` and `DBA_FLASHBACK_TXN_REPORT` views.
- Review the dependency report that shows all transactions that were backed out.
- Commit the changes to make them permanent. *Or:*
- Roll back to discard the changes.



Copyright © 2020, Oracle and/or its affiliates.

The Flashback Transaction Wizard in Enterprise Manager calls the `DBMS_FLASHBACK.TRANSACTION_BACKOUT` procedure with the `NOCASCADE` option.

Usage Notes

- Transaction back-out is not supported across conflicting DDL.
- Transaction back-out inherits data type support from LogMiner.
- When you discover the need for transaction back-out, performance is better if you start the back-out operation sooner. Large redo logs and high transaction rates result in slower transaction back-out operations.
- Provide a transaction name for the back-out operation to facilitate later auditing. If you do not provide a transaction name, it will be automatically generated for you.

The `DBA_FLASHBACK_TXN_STATE` view contains the current state of a transaction—whether it is alive in the system or effectively backed out. This table is atomically maintained with the compensating transaction. For each compensating transaction, there could be multiple rows, where each row provides the dependency relationship between the transactions that have been compensated by the compensating transaction.

The `DBA_FLASHBACK_TXN_REPORT` view provides detailed information about all compensating transactions that have been committed in the database. Each row in this view is associated with one compensating transaction.

Best Practices: Undo-Based Flashback Query, Flashback Table

- Use Undo Advisor in Enterprise Manager to obtain recommendations on available undo retention for various sizes.
- Use fixed size undo: Undo retention is automatically tuned for the best possible retention based on tablespace size and current system load.
- Be aware of DDL restrictions: Not possible to query in the past if table structure is modified.

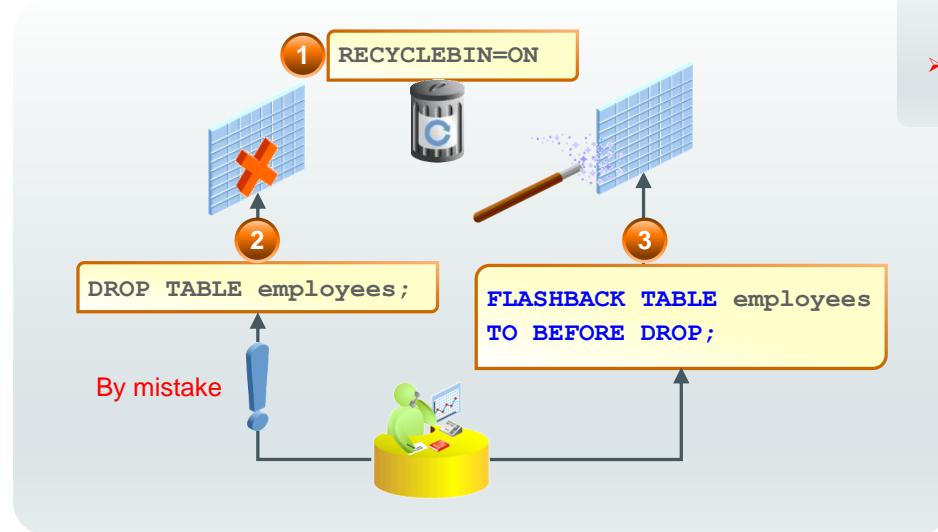


Copyright © 2020, Oracle and/or its affiliates.

Some DDL commands that alter the structure of a table invalidate any existing undo data for the table. Commands that drop and modify columns, move tables, drop partitions, and truncate table/partition fall into this category. It is not possible to retrieve data from a point before the execution of these DDL commands. An attempt to do so results in an ORA-1466 error. Note that this restriction does not apply to DDL operations that alter the storage attributes of a table, such as PCTFREE, INITTRANS, and MAXTRANS.

Flashback Drop and the Recycle Bin

Flashback
 - Query
 - Versions
 - Table
 - Transaction
 ➤ - Drop
 - Data Archive



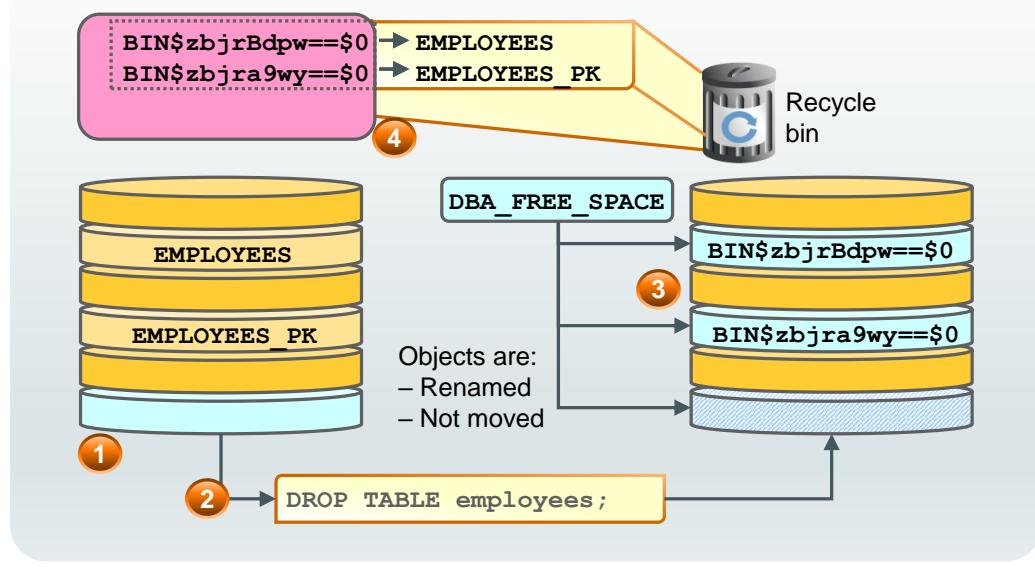
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Using the `FLASHBACK TABLE` command, you can undo the effects of a `DROP TABLE` statement without having to use point-in-time recovery (shown in the graphic in the slide).

1. The `RECYCLEBIN` initialization parameter is used to control whether the Flashback Drop capability is turned `ON` or `OFF`. If the parameter is set to `OFF`, then dropped tables do not go into the recycle bin.
2. If this parameter is set to `ON` (default), the dropped tables go into the recycle bin.
3. Dropped tables can be recovered with the `FLASHBACK TABLE ...TO BEFORE DROP` command.

Recycle Bin



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

If the recycle bin **is not** enabled and you drop a table, the space can immediately be used for other objects. If the recycle bin **is** enabled and you drop a table, the space associated with the table and its dependent objects **is not** immediately reclaimable, even though it does appear in DBA_FREE_SPACE. Instead, the dropped objects are referenced in the recycle bin and still belong to their owner. The space used by recycle bin objects is never automatically reclaimed unless there is space pressure. This enables you to recover recycle bin objects for the maximum possible duration.

When a dropped table is “moved” to the recycle bin, the table and its associated objects and constraints are renamed using system-generated names. The renaming convention is: BIN\$unique_id\$version, where unique_id is a 26-character globally unique identifier for this object making the recycle bin name unique across all databases and version is a version number assigned by the database.

The recycle bin itself is a data dictionary table that maintains the relationships between the original names of dropped objects and their system-generated names. You can query the recycle bin by using the DBA_RECYCLEBIN view. The diagram illustrates this behavior:

1. You have created a table called EMPLOYEES in your tablespace.
2. You drop the EMPLOYEES table.
3. The extents occupied by EMPLOYEES are now considered as free space.
4. EMPLOYEES is renamed and the new name is recorded into the recycle bin.

Bypassing the Recycle Bin

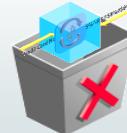
```
DROP TABLE <table_name> [PURGE] ;
```

```
DROP TABLESPACE <ts_name>  
[INCLUDING CONTENTS] ;
```

```
DROP USER <user_name> [CASCADE] ;
```

Security considerations for the recycle bin:

```
ALTER SYSTEM SET RECYCLEBIN=OFF SCOPE=SPFILE;
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

- You can use the `DROP TABLE PURGE` command to permanently drop a table and its dependent objects from the database. When you use this command, the corresponding objects are not moved to the recycle bin.
- When you issue the `DROP TABLESPACE ... INCLUDING CONTENTS` command, the objects in the tablespace are not placed in the recycle bin. Moreover, objects in the recycle bin belonging to the tablespace are purged. When you issue the same command without the `INCLUDING CONTENTS` clause, the tablespace must be empty for the command to succeed. However, there can be objects belonging to the tablespace in the recycle bin. In this case, these objects are purged.
- When you issue the `DROP USER ... CASCADE` command, the user and all the objects owned by the user are permanently dropped from the database. Any objects in the recycle bin belonging to the dropped user are purged.

For increased security, you may decide to not allow the use of the recycle bin (for example, if the current object is encrypted, but the dropped object is in clear text, potentially showing sensitive data). Connected as SYSDBA, you can:

- View the recycle bin status with: `SHOW PARAMETER RECYCLEBIN`
- Disable the use of the recycle bin with: `ALTER SYSTEM SET RECYCLEBIN=OFF SCOPE=SPFILE;` after issuing this command, you need to restart the database

Using Flashback Data Archives

Automated tracking of historical database changes:

- Enable at the table level with your specified retention period.
- All subsequent changes are transparently stored and tamper proof.
- Records older than retention period are automatically removed.
- Use Flashback technologies to retrieve history.

Flashback
- Query
- Versions
- Table
- Transaction
- Drop
➤ - Data Archive

```
SELECT ... AS OF TIMESTAMP...
SELECT ... VERSIONS BETWEEN TIMESTAMP and TIMESTAMP...
```



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Flashback data archives provide a mechanism for tracking changes to production databases, which is secure, efficient, easy to use, and application transparent.

With this technology, you can automatically track and store the data in tables enabled for flashback data archive (FDA). This ensures that flashback queries obtain SQL-level access to the versions of database objects without getting a snapshot-too-old error.

A flashback data archive provides the ability to track and store all transactional changes to a “tracked” table over its lifetime. It is no longer necessary to build this intelligence into your application. You can use this technology for compliance, audit reports, data analysis, and decision-support systems.

Creating a Temporal History and Enabling Archiving

1. Create a new tablespace to hold the history data.
2. Create a flashback data archive, assign it to the tablespace, and specify its retention period. (It requires the FLASHBACK ARCHIVE ADMINISTER system privilege.)

```
CREATE FLASHBACK ARCHIVE fda1 TABLESPACE fda_tbs1  
OPTIMIZE DATA QUOTA 10M RETENTION 1 YEAR;
```

3. Alter the base tables to enable archiving and assign it to a flashback archive. (It requires the FLASHBACK ARCHIVE object privilege.)

```
ALTER TABLE HR.EMPLOYEES FLASHBACK ARCHIVE fda1;
```



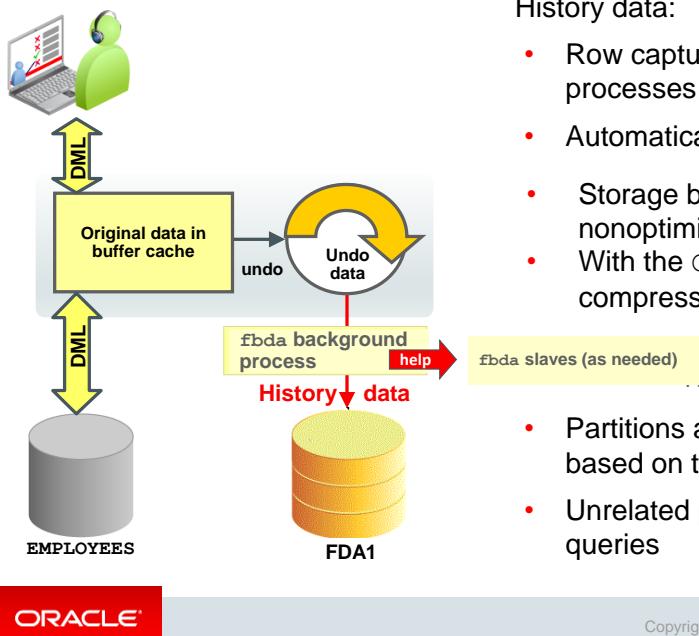
ORACLE

Copyright © 2020, Oracle and/or its affiliates.

A flashback data archive consists of one or more tablespaces. You can have multiple flashback data archives. They are configured with retention duration. Based on your retention duration requirements, you should create different archives—for example, one for all records that must be kept for two years, another for all records that must be kept for five years. The database server will automatically purge all historical information on the day after the retention period expires.

1. Create a tablespace for your FDA. The size depends on the base table and the expected DML and DDL activity.
2. Create a flashback data archive with retention time, by default, with duplication and without compression. Data archived in FDA is retained for the retention time. This task requires the FLASHBACK ARCHIVE ADMINISTER system privilege. If different retention periods are needed, different archives must be created. With the OPTIMIZE DATA clause, the flashback data archive is created with compression and deduplication.
3. Enable the flashback archiving (or disable it) for a (whole) table. This task requires the FLASHBACK ARCHIVE object privilege. Although flashback archiving is enabled for a table, some DDL statements are not allowed on that table. By default, flashback archiving is off for any table.

How the Flashback Data Archive Works



History data:

- Row captured asynchronously by background processes at self-tuned intervals (default: 5 min)
- Automatically purged per retention policy
- Storage by default nonoptimized
- With the `OPTIMIZE DATA` clause: compressed and deduplicated
- Partitions automatically created based on time and volume
- Unrelated partitions skipped by queries

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

History data is captured from undo (and buffer cache) by the `fbda` background process at self-tuned intervals. The default is every five minutes. The entire base table row that is updated is stored, no matter how many columns are updated.

- With the `OPTIMIZE DATA` clause, table and LOB compression and LOB deduplication are automatically turned on, using any of the following features: Advanced Row table compression, SecureFiles Intelligent Compression, SecureFiles Intelligent Deduplication, and segment-level and row-level ILM compression. ILM is enabled to allow new data to be archived as uncompressed and over time is compressed in the background.

Note: If the base table is compressed with Hybrid Columnar compression, the table cannot be enabled for flashback data archiving.

- FDA history tables already compressed and deduplicated in releases prior to 12.1 are not changed. Their storage continues compressed and deduplicated.
- To stop optimization on FDA history tables, execute the following statement:
`SQL> ALTER FLASHBACK ARCHIVE fla1 NO OPTIMIZE DATA;`
- Each flashback archive partition is at least 1 day and 1 MB of data, partitioned on ENDSCN. Flashback queries to the archives avoid unrelated partitions.
- Up to 10 flashback archiver slaves can be called upon by the `fbda` process.
- If the flashback archive process and slaves are too busy, archiving may be performed inline, which significantly affects the user's response time.

Collecting User Context in Temporal History

- Collection level: NONE (default), TYPICAL, ALL
 - Set with DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL
 - TYPICAL collecting database user ID, global user ID, client identifier, service name, module name, and host name
- Obtained from GET_SYS_CONTEXT (USERENV)

```
SQL> EXEC DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL('TYPICAL')
PL/SQL procedure successfully completed
```

```
SQL> SELECT DBMS_FLASHBACK_ARCHIVE.GET_SYS_CONTEXT
  2      (VERSIONS_XID, 'USERENV', 'SESSION_USER'),
  3      VERSIONS_XID, VERSIONS_STARTTIME, VERSIONS_ENDTIME,
  4      employee_id, salary
  5  FROM hr.employees VERSIONS BETWEEN SCN MINVALUE
  6                                AND          MAXVALUE ;
```



Copyright © 2020, Oracle and/or its affiliates.

The user context of a transaction executed on a table with Temporal History is collected and retrievable. The parameters of the USERENV namespace describe the current session. The user context is obtained from DBMS_FLASHBACK_ARCHIVE.GET_SYS_CONTEXT.

The user context collection is controlled by a parameter, set by DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL whose values can be NONE, TYPICAL, or ALL. By default, no user context is collected.

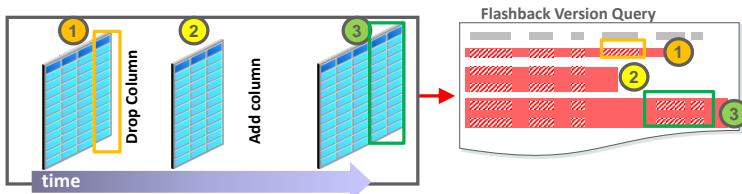
Some examples of information, which are collected in the TYPICAL case, are database user ID, global user ID, client identifier, service name, module name, or host name.

Rows in these tables are purged when the commit time is older than the retention of the flashback archive with the longest retention.

Each row of the user context can be read only by the DBA or the owner of the transaction.

Transparent Schema Evolution

- DDL support for:
 - Add, drop, rename, and modify column
 - Drop and truncate partition
 - Rename and truncate table



- Flashback **queries work across the preceding DDL changes.**
- All other DDL changes are *not* automatically supported.

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

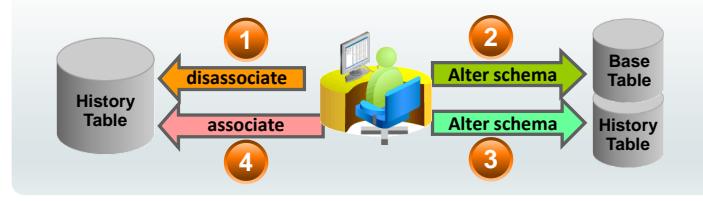
The most common DDL commands are possible with flashback data archives. When a schema has evolved in any of the ways listed in the slide, Temporal History technology automatically keeps track of the changes. Flashback query appropriately returns the row or rows with the corresponding schema (as shown in the diagram).

Full Schema Evolution

Disassociate or associate procedures in the DBMS_FLASHBACK_ARCHIVE package:

- Disable Flashback Archive on specified tables and allow more complex DDL (upgrades, split tables, and so on).
- Enforce schema integrity during association. (Base table and history table must be the same schema.)

Note: This function should be used with care and with the understanding that the archive **can no longer be guaranteed to be immutable** because the history could have been altered during the time of disassociation.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

All DDL changes that are not automatically supported can be executed through the DBMS_FLASHBACK_ARCHIVE package.

Note: This function should be used with care and with the understanding that the archive can no longer be guaranteed to be immutable, because the history could have been altered during the time of disassociation. The system catalog has a note when the disassociation occurred.

The diagram in the slide shows the following workflow:

1. If you have the FLASHBACK_ARCHIVE ADMINISTER privilege, you can disassociate the archive from the base table with the DISASSOCIATE_FBA procedure.
2. Make the necessary changes to the base table.
3. Make the necessary changes to the corresponding archive.
4. Then associate the table with the archive within the same schema with the RESASSOCIATE_FBA procedure. Temporal History validates that the schemas are the same upon association.

There is no transportability of history tables. Some DDL statements cause error ORA-55610 when used on a table enabled for FDA, for example:

- ALTER TABLE ...with UPGRADE_TABLE clause
- ALTER TABLE statement that moves or exchanges a partition or subpartition operation
- DROP TABLE statement

Temporal Validity and History

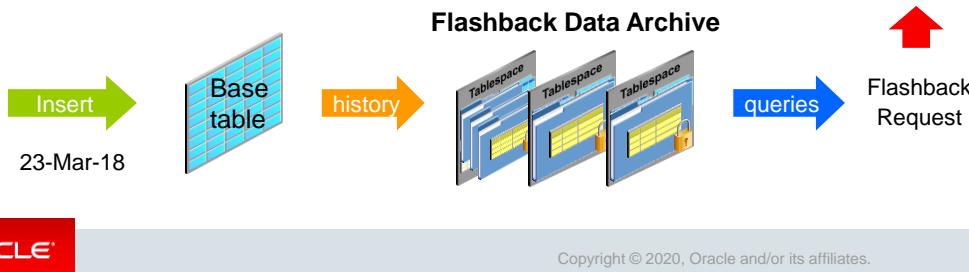
Distinguish active from nonactive rows:

- **Temporal validity:** User-managed effective date in the same table
- **Temporal History:** System-managed transaction time in a separate tablespace

Flashback

- Query
- Versions
- Table
- Transaction
- Drop
- - Data Archive

Emp ID	Job	Hire Date
100	Clerk	22-Apr-11
200	Developer	12-Dec-11
400	Salesman	22-Mar-12



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Applications often annotate the validity of a fact recorded in a table with dates or time stamps that are relevant to the underlying business; for example, the hire date of an employee in HR applications and the effective date of coverage in the insurance industry. This temporal attribute is called temporal validity and is usually controlled by the user who defines the valid-time dimension at table creation.

Temporal validity dates or time stamps are different from the dates or time stamps annotated when the fact was recorded in the database. The date or time stamp when the fact was recorded in the database are attributes of Temporal History (also known as flashback data archive) and are system managed.

In the example in the slide, employee 400 was hired on March 22, but the row was entered in the HR.EMP table on March 23. 22-MAR-12 is the valid time temporal date and 23-MAR-12 is the transaction time temporal date.

By using the valid time temporal implicit filter on the valid-time dimension, queries can show rows that are currently valid or that will be valid in the future. The query is able to hide rows whose facts are not currently valid.

Bi-temporal queries can use both valid-time temporal and transaction time temporal date.

Using the PERIOD FOR Clause

- Keep both active and nonactive data in the same table.
- Define one valid-time dimension at table creation.
 - Explicitly define the two date-time columns.
 - *Or:* automatic valid-time columns are created.

```
SQL> CREATE TABLE emp
  2      ( empno number, salary number, deptid number,
  3           name VARCHAR2(100),
  4           user_time_start DATE, user_time_end DATE,
  5   PERIOD FOR user_time (user_time_start,user_time_end));
```

- Insert rows by explicitly naming the valid-time columns.

```
SQL> CREATE TABLE emp2
  2      ( empno number, salary number, deptid number,
  3           name VARCHAR2(100),
  3   PERIOD FOR user_time);
```



Copyright © 2020, Oracle and/or its affiliates.

A valid-time dimension, represented by the PERIOD FOR clause, consists of two date-time columns that can be specified in the table definition, as shown in the first example, or that are automatically created, as shown in the second example.

To hide valid-time dimension columns, just specify a PERIOD FOR clause name without any date columns. The database server creates two hidden columns using the name of the valid-time dimension as a prefix for the names of the two columns. The valid-time dimension name is used to drop the dimension if required. As shown in the second example, you defined `USER_TIME` as the name of the valid-time dimension, and `USER_TIME` is used as the prefix for the two date columns automatically created: `USER_TIME_START` and `USER_TIME_END`.

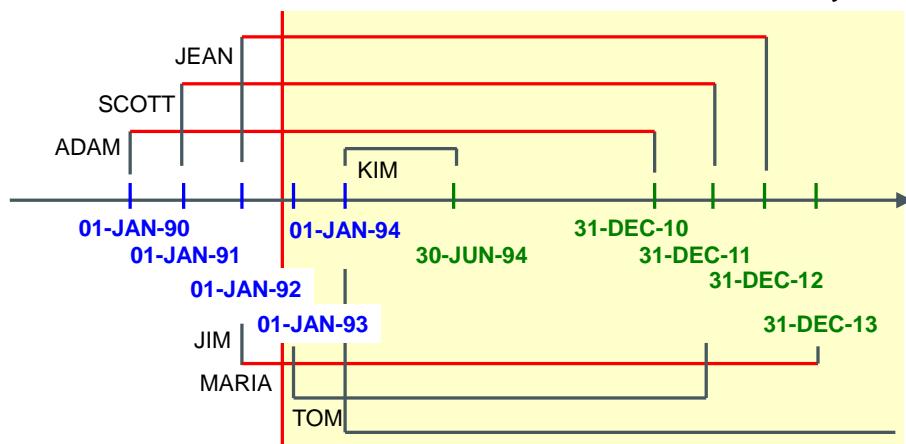
Note: A check constraint is automatically added to ensure that the end time is later than the start time.

To insert rows into a table with a valid-time dimension, you name the two valid-time date columns as follows:

```
SQL> INSERT INTO emp2 (empno, salary, deptid, name, user_time_start,
user_time_end) VALUES (1,1000,20, 'John', SYSDATE, NULL);
SQL> select EMPNO, user_time_start, user_time_end from emp2;
EMPNO USER_TIME_START                      USER_TIME_END
----- -----
1    17-AUG-12 09.58.03.000000 AM +00:00
```

Filtering on Valid-Time Columns: Example 1

Filter on valid-time columns to access active data only



```
SQL> select * from hr.emp as of PERIOD FOR user_time
2          to_date('01-DEC-1992', 'dd-mon-yyyy') ;
```

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

How do you filter on valid-time columns? Use the `SELECT` statement with the `PERIOD FOR` clause or use the `DBMS_FLASHBACK_ARCHIVE` procedure.

- There is one set of data that is “valid” based on its valid-start and valid-end times and the query time (AS OF or undecorated).
- Alternatively, there is the other set of rows where the query time falls outside the valid-start and valid-end times.

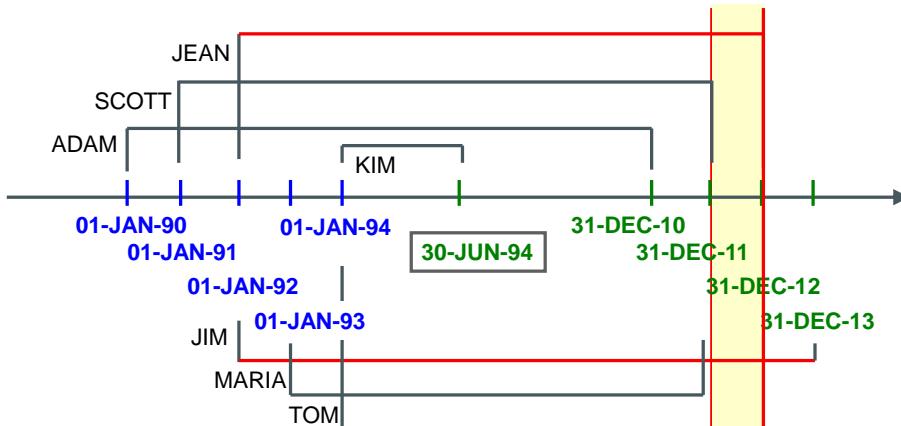
Both sets of rows data reside in the same table. However, by controlling the visibility of data to the valid rows, you can limit what queries and DMLs affect. Until now, you could do as-of and versions queries for transaction time. Now you can do as-of and versions queries for valid time.

For each new employee that you inserted in the table, you included the hire dates, valid-time start dates, and valid end dates. The dates represent the activeness of each row. These dates are entered by the application and correspond to valid dates. The time that the rows were inserted and committed in the table corresponds to the transaction date.

You can filter the active employees by using the following new `PERIOD FOR` clause. The query displays all active employees who were valid at the explicit date of '01-DEC-1992', which is the date that belongs to the valid period, that is, between `USER_TIME_START` and `USER_TIME_END`.

Filtering on Valid-Time Columns: Example 2

Use versions queries for valid-time.



```
SQL> SELECT * FROM hr.emp VERSIONS PERIOD FOR user_time
  2  BETWEEN to_date('31-DEC-2011','DD-MON-YYYY')
  3  AND      to_date('31-DEC-2012','DD-MON-YYYY');
```

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

You can filter the active employees by using the VERTICALS PERIOD FOR BETWEEN clause:

```
select * from hr.emp VERSIONS PERIOD FOR user_time
BETWEEN to_date('31-DEC-2011', 'dd-mon-yyyy')
AND      to_date('31-DEC-2012', 'dd-mon-yyyy') ;
```

The query displays all employees whose VALID_TIME_START is less than or equal to '31-DEC-2011' and VALID_TIME_END greater than or equal to '31-DEC-2012'.

Queries that mix valid-time and transaction-time dimensions are called "bi-temporal queries."

The example shows rows as of the specified transaction time that are valid now.

```
select * from hr.emp
as of period for user_time to_date('31-DEC-1992', 'dd-mon-yyyy')
as of timestamp to_date ('30-mar-2012','dd-mon-yyyy');
```

Using DBMS_FLASHBACK_ARCHIVE

- Visibility control applies to queries and DML.
- Full visibility applies to DDL.
- Visibility set with DBMS_FLASHBACK_ARCHIVE:
 - Set the visibility to data valid as of the given time.

```
DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('ASOF',
                                             (to_timestamp('29-SEP-10 05.44.01 PM')))
```

- Set the visibility to data currently valid within the valid time period at the session level.
- Set the visibility to data to the full table level.

```
DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('CURRENT')
```

```
DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('ALL')
```



Copyright © 2020, Oracle and/or its affiliates.

Users can modify visibility within a session by using the new DBMS_FLASHBACK_ARCHIVE package. Visibility control applies to all SQL SELECT and DML statements.

DDLS will default to getting full visibility to the table data. For example, CTAS, online redefinition, and ALTER TABLE MOVE operations will have full visibility of the table data. Hidden columns are also visible to the DDL operations, resulting in preservation of those columns and their data.

The first example sets the valid time visibility to '29-SEP-2010', showing only rows overlapping the given date.

The second example sets the visibility to data currently valid within the period time at the session level.

The third example sets the visibility to the full table, which is the default temporal table visibility.

Summary

In this lesson, you should have learned how to:

- Use flashback technologies to protect against and recover from various types of errors
- Perform flashback operations
- Distinguish between temporal validity and temporal history



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Preparing to Use Flashback Technologies
- Restoring a Dropped Table
- Using Flashback Table



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Using Flashback Database

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Describe the Flashback Database architecture
- Configure your database to support Flashback Database
- Perform the Flashback Database operation



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Preparing Your Database for Flashback

- Grant FLASHBACK privileges.
- Relevant undo settings:
 - UNDO_TABLESPACE='UNDOTBS1'
 - UNDO_MANAGEMENT='AUTO'
 - UNDO_RETENTION=900
 - Guaranteeing undo retention



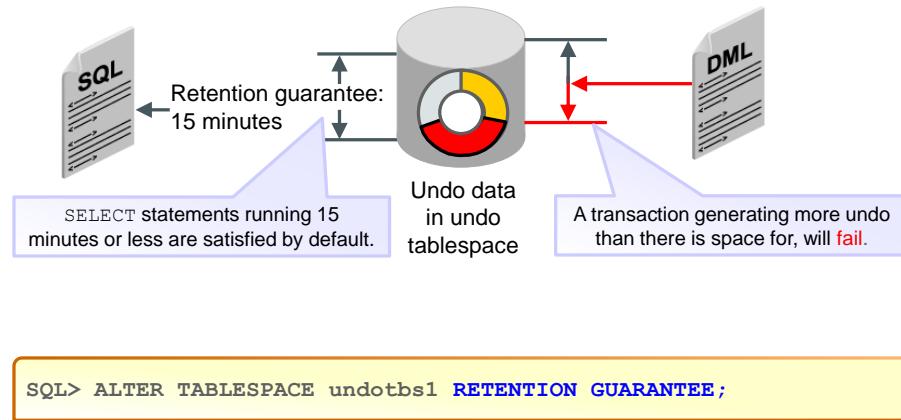
Copyright © 2020, Oracle and/or its affiliates.

To enable flashback features for an application, you must:

- Grant flashback privileges to users, roles, or applications that need to use flashback features
- Have an undo tablespace with enough space to keep the required data for flashback operations. The more often users update the data, the more space is required.

For an automatically extensible undo tablespace (default), the Oracle database retains undo data to satisfy at a minimum, the retention periods needed by the longest-running query and the threshold of undo retention, specified by the `UNDO_RETENTION` parameter. You can query `V$UNDOSTAT.TUNED_UNDORETENTION` to determine the amount of time for which undo is retained for the current undo tablespace. Setting the `UNDO_RETENTION` parameter does not guarantee that unexpired undo data is not overwritten. The default undo behavior is to overwrite committed transactions that have not yet expired rather than to allow an active transaction to fail because of lack of undo space. In case of conflict, transactions have precedence over queries. This behavior can be changed by guaranteeing retention.

Guaranteeing Undo Retention



```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

With guaranteed retention, undo retention settings are enforced even if they cause transactions to fail. (So in case of conflict, queries have precedence over transactions, as shown in the graphic in the slide.)

Specify the `RETENTION GUARANTEE` clause for the undo tablespace to ensure that unexpired undo data is not discarded. Note that by configuring retention guarantee, ongoing operations that need undo space in the segments of the tablespace may fail due to lack of space. `RETENTION GUARANTEE` is a tablespace attribute rather than an initialization parameter. An example is shown in the slide. To return a guaranteed undo tablespace to its normal setting:

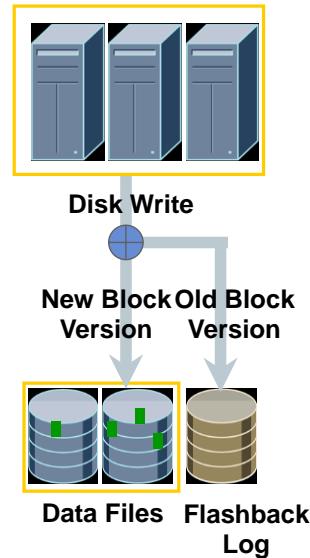
```
SQL> ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;
```

To satisfy long retention requirements, create a Temporal History.

Flashback Database: Continuous Data Protection

- Is a fast point-in-time recovery strategy
- Eliminates the need to restore a whole database backup
- Provides continuous data protection for the database
- Optimized: Before-change block logging
- Restores just changed blocks
- Replays log to restore the database to the desired time
- Provides fast recovery: Minutes, not hours
- Requires a single command to restore:

```
FLASHBACK DATABASE TO '2:05 PM'
```



Copyright © 2020, Oracle and/or its affiliates.

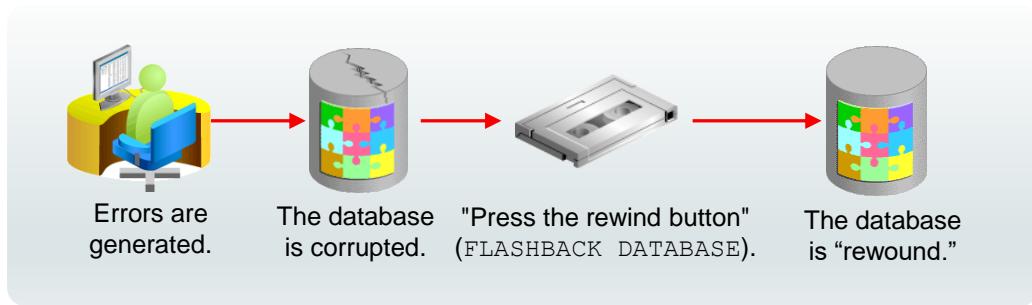
Flashback Database is a unique database point-in-time recovery capability, which enables the database to be quickly “rewound” to a previous point in time. Flashback Database restores the database more quickly than traditional restore and recovery methods, because only the affected data blocks are restored and recovered.

Flashback Database uses flashback logs, which record old block versions. The graphic in the slide illustrates how the old block version is written to the flashback log and the new block version is written to the data file when writes are issued to disk and Flashback Database is enabled. When the FLASHBACK DATABASE command is issued, only the changed blocks are retrieved from the flashback logs. The blocks are then recovered with the appropriate archived logs to the required point in time.

Flashback Database

The Flashback Database operation:

- Works like a rewind button for the database
- Can be used in cases of logical data corruptions made by users



ORACLE®

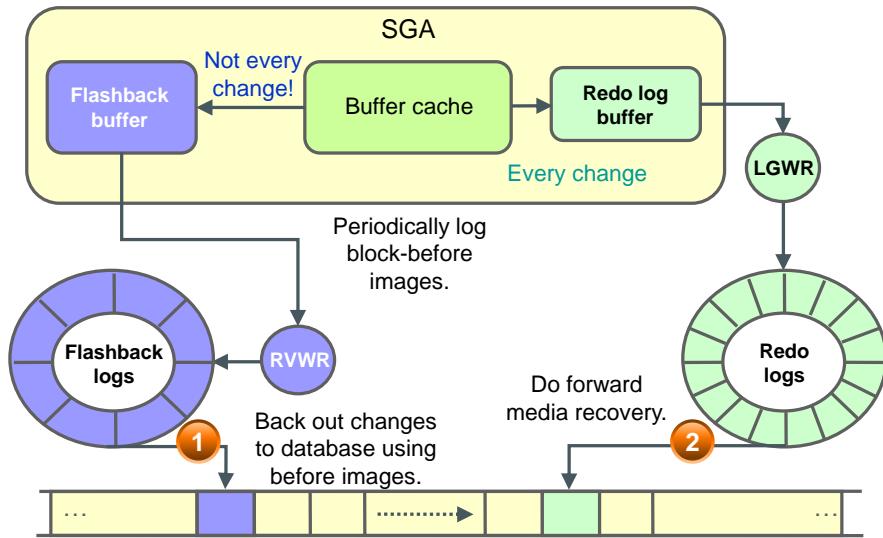
Copyright © 2020, Oracle and/or its affiliates.

With Flashback Database, you can quickly bring your database to an earlier point in time by undoing all the changes that have taken place since that time. This operation is fast because you do not need to restore backups. You can use this feature to undo changes that have resulted in logical data corruptions.

When you use Flashback Database, Oracle Database server uses past block images to back out changes to the database. During normal database operation, Oracle Database server occasionally logs these block images in flashback logs. Flashback logs are written sequentially and are not archived. Oracle Database server automatically creates, deletes, and resizes flashback logs in the fast recovery area. You need to be aware of flashback logs only for monitoring performance and deciding how much disk space to allocate for them in the fast recovery area.

The time it takes to rewind a database with Flashback Database is proportional to how far back in time you need to go and the amount of database activity after the target time. The time it would take to restore and recover the whole database could be much longer. The before images in the flashback logs are used only to restore the database to a point in the past, and forward recovery is used to bring the database to a consistent state at some time in the past.

Flashback Database Architecture



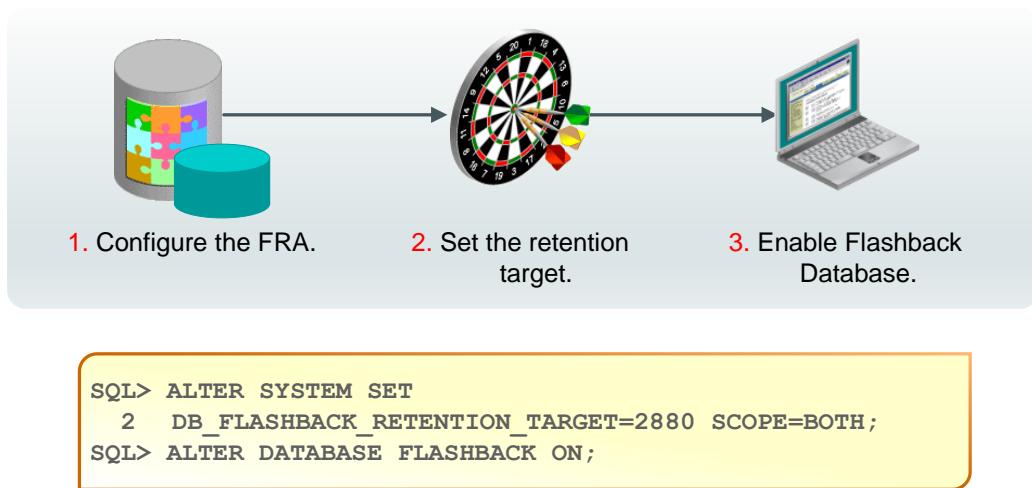
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

When you enable Flashback Database, the **RVWR** (Flashback Writer) background process is started. This background process sequentially writes Flashback Database data from the flashback buffer to the Flashback Database logs, which are circularly reused. Subsequently, when a **FLASHBACK DATABASE** command is issued, the flashback logs are used to restore to the blocks' before images, and then redo data is used to roll forward to the desired flashback time.

The overhead of enabling Flashback Database depends on the read/write mix of the database workload. Because queries do not need to log any flashback data, the more write-intensive the workload, the higher the overhead of turning on Flashback Database.

Configuring Flashback Database



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

You can configure Flashback Database as follows:

1. Configure the fast recovery area (FRA).
2. Set the retention target with the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter. You can specify an upper limit, in minutes, on how far back you want to be able to flashback the database. The example uses 2,880 minutes, which is equivalent to two days. This parameter is only a target and does not provide any guarantee. Your flashback time interval depends on how much flashback data has been kept in the fast recovery area.
3. Enable Flashback Database with the following command:

```
ALTER DATABASE FLASHBACK ON;
```

Before you can issue the command to enable Flashback Database, the database must be configured for archiving. You can enable Flashback Database when the database is open.

You can determine whether Flashback Database is enabled with the following query:

```
SELECT flashback_on FROM v$database;
```

You can disable Flashback Database with the `ALTER DATABASE FLASHBACK OFF` command. As a result, all existing Flashback Database logs are deleted automatically.

Flashback Database: Examples

- To flash back: Mount (in exclusive mode) the database.

```
RMAN> FLASHBACK DATABASE TO TIME =
2> "TO_DATE('2009-05-27 16:00:00',
3> 'YYYY-MM-DD HH24:MI:SS')";

RMAN> FLASHBACK DATABASE TO SCN=23565;

RMAN> FLASHBACK DATABASE
2> TO SEQUENCE=223 THREAD=1;

SQL> FLASHBACK DATABASE
2> TO TIMESTAMP(SYSDATE-1/24);
SQL> FLASHBACK DATABASE TO SCN 53943;
SQL> FLASHBACK DATABASE TO RESTORE POINT b4_load;
```

- To review changes: Open in read-only mode.
- To finalize: Open in read/write mode with RESETLOGS.



Copyright © 2020, Oracle and/or its affiliates.

You can use the RMAN FLASHBACK DATABASE command to execute the Flashback Database operation. You can use SEQUENCE and THREAD to specify a redo log sequence number and thread as a lower limit. RMAN selects only files that can be used to flash back to, but not including, the specified sequence number.

Alternatively, you can use the SQL FLASHBACK DATABASE command to return the database to a past time or SCN. If you use the TO SCN clause, you must provide a number. If you specify TO TIMESTAMP, you must provide a time stamp value. You can also specify a restore point name.

You can monitor the Flashback Database progress with the V\$SESSION_LONGOPS view.

Note: The database must be mounted in exclusive mode to issue the FLASHBACK DATABASE command and opened read-only to review changes. The database must be opened read/write with the RESETLOGS option when finished.

CDB and PDB Flashback

- You cannot flash back the CDB root without flashing back the entire CDB.
- PDB flashback is similar to CDB flashback.

```
RMAN> CONN sys@pdb1
RMAN> ALTER PLUGGABLE DATABASE CLOSE;
RMAN> FLASHBACK PLUGGABLE DATABASE pdb1 TO SCN 411010;
RMAN> ALTER PLUGGABLE DATABASE pdb1 OPEN RESETLOGS;
```



Copyright © 2020, Oracle and/or its affiliates.

You cannot flash back the CDB root without flashing back the entire CDB. You can also flash back a PDB.

All datafiles belonging to a PDB can be flashed back and recovered in place. All undo application that is needed to make a PDB consistent after flashback is applied in place. After a PDB flashback operation, the old backup of the PDB is still valid.

In the second example in the slide, `OPEN RESETLOGS` creates a new incarnation for the PDB, similar to a database `OPEN RESETLOGS`, which creates a new incarnation for the CDB. The CDB root must be opened when the PDB `OPEN RESETLOGS` is performed, except that recovering a PDB to a point in time does not affect all parts of the CDB. The whole CDB and all other PDBs are opened. After recovering a PDB to a specified point in time, when you open the PDB by using the `RESETLOGS` option, a new incarnation of the PDB is created.

The PDB `RESETLOGS` option does not perform a `RESETLOGS` for the CDB. A PDB record in the control file is updated. Each redo log record carries the PDB ID in the redo header. This is how recovery knows which redo applies to which PDB. Redo logs are shared by all PDBs; redo from each PDB is written to a single set of redo logs. Conceptually, a PDB `OPEN RESETLOGS` is similar to a database `OPEN RESETLOGS`. Query `V$PDB_INCARNATION` for details.

This is very useful when a user error has been issued, for example, a `DROP USER`. Instead of restoring and recovering the PDB, flashback to the time before the user error had been issued is easy.

Flashback Database Considerations

- When the Flashback Database operation completes, open the database:
 - In read-only mode, to verify that the correct target time or SCN was used
 - With a `RESETLOGS` operation to allow DML
- You cannot use Flashback Database in the following situations:
 - The control file has been restored or re-created.
 - A tablespace has been dropped.
 - A data file has been reduced in size.
- Use the `TO BEFORE RESETLOGS` clause to flash back to before the last `RESETLOGS` operation.



Copyright © 2020, Oracle and/or its affiliates.

In situations where you cannot use the Flashback Database feature, you should use an incomplete recovery operation to return the database to a specific time. After the Flashback Database operation is complete, you can open the database in read-only mode to verify that the correct target time or SCN was used. If not, you can flash back the database again or perform a recovery to roll forward the database.

You cannot use Flashback Database to recover a data file that was dropped during the span of time you are flashing back. The dropped data file is added to the control file and marked offline, but it is not flashed back. Flashback Database cannot flash back a data file to a time after its creation and before the resize operation. If a file was resized during the span of time to which you are going to flash back the database, then you should take the file offline before beginning the Flashback Database operation. This is applicable for files that are shrunk rather than expanded. You can use Flashback Database with data files that you have configured for automatic extension. You can flash back to just before the last `RESETLOGS` operation by supplying the `TO BEFORE RESETLOGS` clause in the `FLASHBACK DATABASE` command.

Note: The flashback retention target is not an absolute guarantee that flashback will be available. If space is needed for required files in the fast recovery area, flashback logs may be deleted automatically.

Monitoring Flashback Database Information

To monitor the ability to meet your retention target:

- View the fast recovery area disk quota:

```
SQL> SELECT estimated_flashback_size,
  2         flashback_size
  3    FROM    V$FLASHBACK_DATABASE_LOG;
```

- Determine the current flashback window:

```
SQL> SELECT oldest_flashback_scn,
  2   oldest_flashback_time
  3  FROM    V$FLASHBACK_DATABASE_LOG;
```

- Monitor logging in the Flashback Database logs:

```
SQL> SELECT *
  2  FROM    V$FLASHBACK_DATABASE_STAT;
```



Copyright © 2020, Oracle and/or its affiliates.

It is important for you to monitor space usage of the fast recovery area so that you know how well you are meeting your retention target. Use the `V$FLASHBACK_DATABASE_LOG` view to monitor the Flashback Database retention target:

- `ESTIMATED_FLASHBACK_SIZE` uses previously logged flashback data to provide an estimate of how much disk space is needed in the fast recovery area for flashback logs to meet the current flashback retention target. The estimate is based on the workload since the instance was started or during the most recent time interval equal to the flashback retention target, whichever is shorter.
- `FLASHBACK_SIZE` gives you the current size, in bytes, of the flashback data.
- `OLDEST_FLASHBACK_SCN` and `OLDEST_FLASHBACK_TIME` display the approximate lowest SCN and time to which you can flash back your database. `CURRENT_SCN` in `V$DATABASE` gives you the current database SCN.

Use the `V$FLASHBACK_DATABASE_STAT` view to monitor the overhead of logging flashback data in the Flashback Database logs. This view contains 24 hours of information, with each row representing a one-hour time interval. You can use this view to determine rate changes in the flashback data generation.

```
SQL> SELECT begin_time, end_time, flashback_data, db_data,
  2  redo_data, estimated_flashback_size AS EST_FB_SZE
  3  FROM V$FLASHBACK_DATABASE_STAT;
```

BEGIN_TIM	END_TIME	FLASHBACK_DATA	DB_DATA	REDO_DATA	EST_FB_SZE
12-FEB-09		16384	0	24576	0
12-FEB-09		6594560	7471104	1533440	815923200
12-FEB-09		17235968	12361728	5150920	839467008
12-FEB-09		311648256	37249024	10272768	855195648

Based on this information, you may need to adjust the retention time or the fast recovery area size.

`FLASHBACK_DATA` and `REDO_DATA` represent the number of bytes of flashback data and redo data written, respectively, during the time interval, and `DB_DATA` gives the number of bytes of data blocks read and written. This view also contains the estimated flashback space needed for the interval.

You can query `V$RECOVERY_FILE_DEST` to view information regarding the fast recovery area. The column descriptions are:

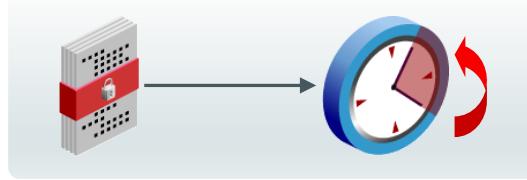
- **NAME:** Fast recovery area name, indicating location string
- **SPACE_LIMIT:** Disk limit specified in the `DB_RECOVERY_FILE_DEST_SIZE` parameter
- **SPACE_USED:** Space used by fast recovery area files (in bytes)
- **SPACE_RECLAIMABLE:** Amount of space that can be reclaimed by deleting obsolete, redundant, and other low-priority files through the space management algorithm
- **NUMBER_OF_FILES:** Number of files

```
SQL> SELECT name, space_limit AS quota,
  2      space_used      AS used,
  3      space_reclaimable AS reclaimable,
  4      number_of_files   AS files
  5  FROM v$recovery_file_dest;
```

NAME	QUOTA	USED	RECLAMABLE	FILES
/u01/flash_recovery_area	5368707120	2507809104	203386880	226

Guaranteed Restore Points

A guaranteed restore point ensures that you can perform a FLASHBACK DATABASE command to that SCN at any time.



```
SQL> CREATE RESTORE POINT before_upgrade  
2 GUARANTEE FLASHBACK DATABASE;
```

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Like normal restore points, guaranteed restore points can be used as aliases for SCNs in recovery operations. A principal difference is that guaranteed restore points never age out of the control file and must be explicitly dropped. However, they also provide specific functionality related to the use of the Flashback Database feature.

Creating a guaranteed restore point at a particular SCN enforces the requirement that you can perform a Flashback Database operation to return your database to its state at that SCN, even if flashback logging is not enabled for your database. If flashback logging is enabled, creating a guaranteed restore point enforces the retention of flashback logs required for Flashback Database back to any point in time after the creation of the earliest guaranteed restore point.

A guaranteed restore point can be used to revert a whole database to a known good state days or weeks ago, as long as there is enough disk space in the fast recovery area to store the needed logs. As with normal restore points, guaranteed restore points can be used to specify a point in time for RECOVER DATABASE operations.

Note: Limitations that apply to Flashback Database also apply to guaranteed restore points. For example, shrinking a data file or dropping a tablespace can prevent flashing back the affected data files to the guaranteed restore point.

Flashback Database and Guaranteed Restore Points

To use guaranteed restore points, the database must satisfy the following prerequisites:

- The database must be in ARCHIVELOG mode.
- FLASHBACK DATABASE requires the use of archived redo logs starting from the time of the restore point.
- A fast recovery area must be configured.



Copyright © 2020, Oracle and/or its affiliates.

To support the use of guaranteed restore points, the database must satisfy the following prerequisites:

- The database must be running in ARCHIVELOG mode.
- To rewind the database to a guaranteed restore point, the FLASHBACK DATABASE command needs the archived redo logs starting from the time of the restore point.
- A fast recovery area must be configured because Oracle Database server stores the required logs in the fast recovery area.
- If Flashback Database is not enabled, the database must be mounted, not open, when creating the first guaranteed restore point (or if all previously created guaranteed restore points have been dropped).

Logging for Flashback Database and guaranteed restore points involves capturing images of data file blocks before changes are applied. The FLASHBACK DATABASE command can use these images to return the data files to their previous state. The main difference between normal flashback logging and logging for guaranteed restore points is whether the logs can be deleted in response to space pressure in the fast recovery area. This difference affects space usage for logs and database performance.

If you enable Flashback Database and define one or more guaranteed restore points, then the database performs normal flashback logging. In this case, the recovery area retains the flashback logs required to flash back to any arbitrary time between the present and the earliest currently defined guaranteed restore point. Flashback logs are not deleted in response to space pressure if they are required to satisfy the guarantee.

PDB Flashback and Clean Restore Point

- Clean PDB restore points can be created after a PDB is closed and ONLY in shared undo mode.
- The benefits of clean PDB restore points include:
 - Faster than other types of PDB flashback
 - No restore of any backup
 - No clone instance created
 - No need to take a new backup

```
SQL> CONNECT / AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;
SQL> CREATE CLEAN RESTORE POINT start_step1 FOR PLUGGABLE DATABASE pdb1
      GUARANTEE FLASHBACK DATABASE;
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN;
SQL> @script_patch_step1
SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

$ rman target /
RMAN> FLASHBACK PLUGGABLE DATABASE pdb1 TO RESTORE POINT start_step1;
RMAN> ALTER PLUGGABLE DATABASE pdb1 OPEN RESETLOGS;
```



Copyright © 2020, Oracle and/or its affiliates.

Normal and Guaranteed Restore Points

PDB normal and guaranteed restore points are restore points that pertain to only a specific PDB and are visible within the PDB only.

- A normal PDB restore point is essentially a bookmark for a past point for that PDB.
- A guaranteed PDB restore point guarantees a PDB flashback to that restore point.

PDB flashback uses PDB point-in-time recovery to recover the shared undo. RMAN automatically restores the shared undo and certain tablespaces in a clone instance and recovers the data to the same point in time.

Clean Restore Points

A clean PDB restore point can be created after a PDB is closed with no outstanding transactions and ONLY in shared undo mode. Thus, flashing back a PDB to a clean restore point is faster than other types of flashback because it does not require restoring backups or creating a clone instance. For example, a DBA expecting to roll back an application upgrade should consider creating a clean PDB guaranteed restore point before the application upgrade.

A restore point that is created while being connected to the CDB root without specifying the new `FOR PLUGGABLE DATABASE` clause is called a CDB restore point.

In case the PDB is the application root of an application container, all restore points are “clean” with local undo, because the availability of local undo means that the effects of active transactions at the time of the restore point can be undone.

Caution: PDB guaranteed restore points can potentially result in Oracle running out of space in the FRA. The DBA should remove PDB guaranteed restore points when the PDB flashback operations are completed.

Best Practices: Flashback Database

- Tune fast recovery area storage.
 - Use ASM, configure enough disk spindles, and so forth.
- Use physical standby database to test flashback logging.
- Use v\$FLASHBACK_DATABASE_LOG to size log space, after running workload that runs longer than the flashback retention period.
- Create guaranteed restore point (GRP) without enabling flashback logging.
 - It saves disk space for workloads where the same blocks are repeatedly updated.
 - Drop GRP to immediately reclaim space.



Copyright © 2020, Oracle and/or its affiliates.

To achieve good performance for large production databases by using Flashback Database, Oracle recommends the following:

- Use a fast file system for your fast recovery area, preferably without operating system file caching. Files that are stored in the fast recovery area, including flashback logs, are typically large. Operating system file caching is typically not effective for these files and may actually add CPU overhead due to reading from and writing to these files. Use a file system such as ASM.
- Configure enough disk spindles for the file system that will hold the fast recovery area. Multiple disk spindles may be needed to support the disk throughput required for Oracle Database server to write the flashback logs effectively.
- If the storage system used to hold the fast recovery area does not have nonvolatile RAM, try to configure the file system on top of striped storage volumes with a relatively small stripe size such as 128 K. This enables each write to the flashback logs to be spread across multiple spindles, thereby improving performance.
- Set the LOG_BUFFER initialization parameter to at least 8 MB. This ensures that Oracle Database server allocates the maximum amount of memory (typically 16 MB) for writing flashback database logs.

Flashback retention should be set to at least 60 minutes. Oracle Database server writes a metadata marker into the flashback logs approximately every 30 minutes. If the flashback retention is set to less than 60 minutes, a needed marker could be deleted if there is space pressure in the flashback log.

A *restore point* is a user-defined name that is associated with a database point in time. It is used with Flashback Database, Flashback Table, and RMAN. A *guaranteed restore point (GRP)* is a special type of restore point that ensures flashback logs are kept until the restore point is used or deleted. When flashback logging is enabled, flashback logs are generally created in proportion to archived redo logs generated during the same retention period. When flashback logging is disabled and a GRP is created, each changed block is logged only once to maintain the GRP. There is continuous logging of blocks when flashback logging is enabled. To save space, you can create a GRP for fast recovery of a specific set of transactions such as a batch job and then delete the GRP to reclaim the space.

Summary

In this lesson, you should have learned how to:

- Describe Flashback Database architecture
- Configure your database to support Flashback Database
- Perform the Flashback Database operation



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Practice Overview

- Enabling Flashback Logging
- Performing Flashback Database



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Using PDB Snapshots

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives

After completing this lesson, you should be able to:

- Create PDB snapshots
- Create PDBs by using PDB snapshots
- Flash back PDBs by using PDB snapshots

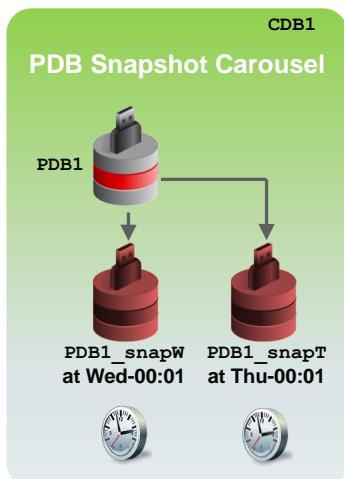


ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Refer to the *Oracle Database Licensing Information User Manual* to determine which Oracle Database editions and services offer the PDB Snapshot Carousel feature.

PDB Snapshot Carousel



A PDB snapshot is a named copy of a PDB at a specific point in time.

- Recovery extended beyond flashback retention period
- Reporting on historical data kept in snapshots
- Storage-efficient snapshot clones taken on periodic basis
- Maximum of eight snapshots for CDB and each PDB
- Example:
 - On Friday, need to recover back to Wednesday
 - Restore PDB1_snapW.

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

A newly created PDB is enabled for PDB snapshots by default, although you can specify whether it is enabled for PDB snapshots or not. A PDB snapshot is an archive file (.pdb) containing the contents of the copy of the PDB at snapshot creation.

PDB snapshots allow the recovery of PDBs back to the oldest PDB snapshot available for a PDB. This feature extends the recovery beyond the flashback retention period that necessitates database flashback enabled.

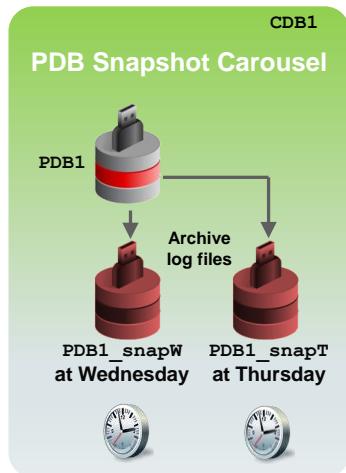
The example in the slide shows a situation where you have to restore PDB1 back to Wednesday.

A use case of PDB snapshots is reporting on historical data. You might create a snapshot of a sales PDB at the end of the financial quarter. You could then create a PDB based on this snapshot so as to generate reports from the historical data.

Every PDB snapshot is associated with a snapshot name and the SCN and timestamp at snapshot creation. The MAX_PDB_SNAPSHOTS database property sets the maximum number of PDB snapshots for each PDB. The default and allowed maximum is 8. When the maximum number is reached for a PDB, and an attempt is made to create a new PDB snapshot, the oldest PDB snapshot is purged. If the oldest PDB snapshot cannot be dropped because it is open, an error is raised. You can decrease this limit for a given PDB by issuing an ALTER DATABASE statement specifying a max number of snapshots. If you want to drop all PDB snapshots, you can set the limit to 0.

Creating PDB Snapshots

1. Enable the PDB for PDB snapshots.



```
SQL> CREATE PLUGGABLE DATABASE pdb1 ...
      SNAPSHOT MODE MANUAL;
```

```
SQL> ALTER PLUGGABLE DATABASE pdb1
      SNAPSHOT MODE EVERY 24 HOURS;
```

2. Create (multiple) manual PDB snapshots of a PDB.

```
SQL> ALTER PLUGGABLE DATABASE pdb1
      SNAPSHOT pdb1_first_snap;
SQL> ALTER PLUGGABLE DATABASE pdb1
      SNAPSHOT pdb1_second_snap;
```

3. Disable snapshot creation for a PDB.

```
SQL> ALTER PLUGGABLE DATABASE pdb1 SNAPSHOT MODE NONE;
```

ORACLE

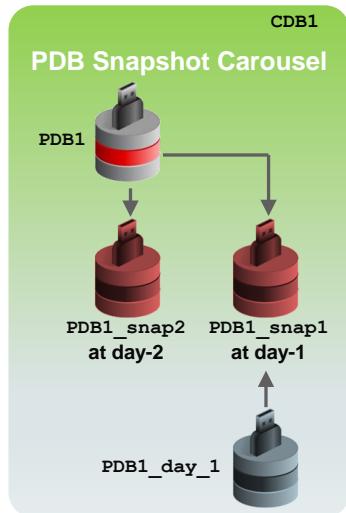
Copyright © 2020, Oracle and/or its affiliates.

By default, a PDB is enabled for PDB snapshots. There are two ways to define PDBs enabled for PDB snapshot creation:

- **Manually:** The first example in the slide uses the SNAPSHOT MODE MANUAL clause of the CREATE PLUGGABLE DATABASE or ALTER PLUGGABLE DATABASE statement. No clause gives the same result.
- **Automatically after a given interval of time:** The second example in the slide uses the SNAPSHOT MODE EVERY <snapshot_interval> [MINUTES|HOURS] clause of the CREATE PLUGGABLE DATABASE or ALTER PLUGGABLE DATABASE statement. When the amount of time is expressed in minutes, it must be less than 3000. When the amount of time is expressed in hours, it must be less than 2000. In the second example in the slide, the SNAPSHOT MODE clause specifies that a PDB snapshot is created automatically every 24 hours.

Every PDB snapshot is associated with a snapshot name and the SCN and timestamp at snapshot creation. You can specify a name for a PDB snapshot. The third and fourth examples in the slide show how to create PDB snapshots manually, even if the PDB is set to have PDB snapshots created automatically. If PDB snapshots are created automatically, the system generates a name.

Creating PDBs by Using PDB Snapshots



After a PDB snapshot is created, you can create a new PDB from it:

```
SQL> CREATE PLUGGABLE DATABASE pdb1_day_1 FROM pdb1  
USING SNAPSHOT <snapshot_name>;
```

```
SQL> CREATE PLUGGABLE DATABASE pdb1_day_2 FROM pdb1  
USING SNAPSHOT AT SCN <snapshot_SCN>;
```



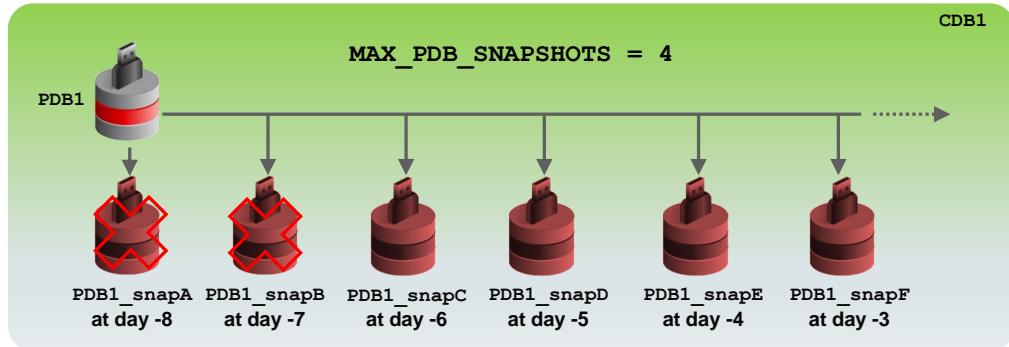
Copyright © 2020, Oracle and/or its affiliates.

You can create a new PDB from an existing PDB snapshot by using the USING SNAPSHOT clause. Provide any of the following:

- The snapshot name
- The snapshot SCN at which the snapshot was created
- The snapshot time at which the snapshot was created

Dropping PDB Snapshots

- Automatic PDB snapshot deletion when MAX_PDB_SNAPSHOTS limit is reached:



- Manual PDB snapshot deletion:

```
SQL> ALTER PLUGGABLE DATABASE pdb1 DROP SNAPSHOT pdb1_first_snap;
```

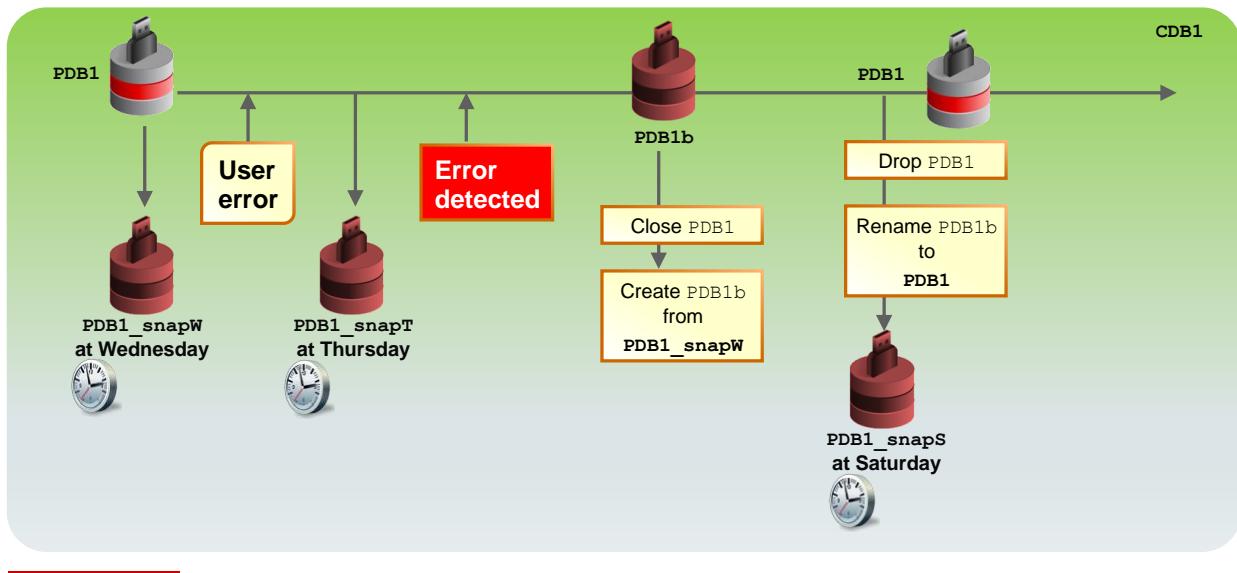
ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

When the carousel reaches eight PDB snapshots or the maximum number of PDB snapshots defined, the oldest PDB snapshot is deleted automatically, whether or not it is in use. There is no need to materialize a PDB snapshot in carousel, because PDB snapshots are all full clone. Be aware that if the SNAPSHOT COPY clause is used with the USING SNAPSHOT clause, the SNAPSHOT COPY clause is simply ignored.

You can manually drop the PDB snapshots by altering the PDB for which the PDB snapshots were created and by using the DROP SNAPSHOT clause.

Flashing Back PDBs by Using PDB Snapshots



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

PDB snapshots enable the recovery of PDBs back to the oldest PDB snapshot available for a PDB.

The example in the slide shows a situation where you detect an error that happened between PDB1_SNAPW and PDB1_SNAPT creation. To recover, perform the following steps:

1. Close PDB1.
2. Create PDB1b from the PDB1_SNAPW snapshot created before the user error.
3. Drop PDB1.
4. Rename PDB1b to PDB1.
5. Open PDB1 and create a new snapshot.

Summary

In this lesson, you should have learned how to:

- Create PDB snapshots
- Create PDBs by using PDB snapshots
- Flash back PDBs by using PDB snapshots



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

Database Duplication Overview



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- List the purposes of creating a duplicate database
- Choose a technique for duplicating a database

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Using a Duplicate Database

- Use a duplicate database to:
 - Test backup and recovery procedures
 - Recover objects by creating an export and importing the objects into the production database
- Create a duplicate database:
 - With the RMAN DUPLICATE command
 - On the same or separate hosts
 - With the identical content or subset of source
 - Performed by auxiliary channels for backup-based duplication



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

A duplicate database is a copy of your target database. With the FOR STANDBY clause, it keeps the same unique database identifier (DBID); if FOR STANDBY is not specified, it creates a new DBID.

You can operate it independently of the target database to:

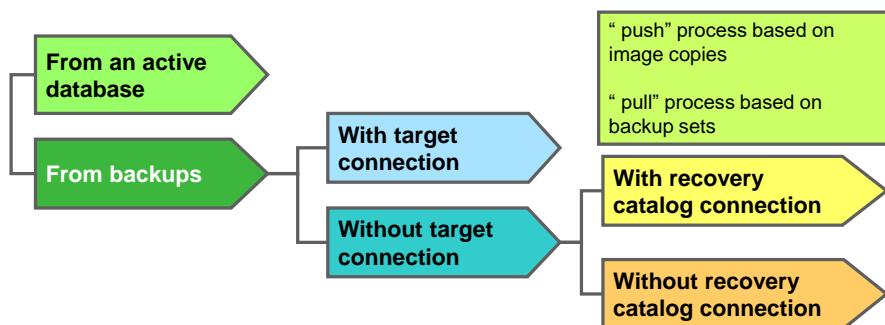
- Test backup and recovery procedures
- Recover objects that were inadvertently dropped from the target database by creating an export containing the objects in the duplicate database and importing them into the production database, although you will probably find that Flashback Query, Flashback Drop, Flashback Table, and table recovery from backup are much easier and faster solutions to recover objects.

To create a duplicate database, you can use the RMAN DUPLICATE command.

- The duplicate database can include the same content or only a subset from the source database. It can be in the same host or a separate host.
- The principal work of the duplication is performed by the auxiliary channels. These channels correspond to a server session on the auxiliary instance on the destination host for backup-based duplication.
- For active database duplication, the target channels perform the work of pushing data file copies to the auxiliary instance (if number of allocated target channels is greater than the number of allocated auxiliary channels).

Choosing Database Duplication Techniques

Choosing a technique to duplicate your database—always with connection to the auxiliary instance:



Via RMAN command line or Enterprise Manager Cloud Control



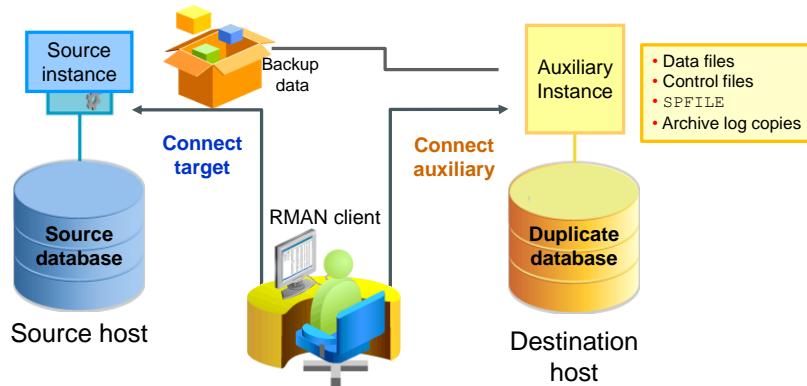
Copyright © 2020, Oracle and/or its affiliates.

You can duplicate a source database to a destination database, which can be on the same or different computers. The database instance associated with the duplicate database is called the auxiliary instance. All duplication techniques require a connection to the auxiliary instance. The diagram shows you the following techniques for database duplication:

- From an active database, connected to the target and auxiliary instances, it is a “pull” (or restore) process based on backup sets.
- From backup, connected to the target and auxiliary instances
- From backup, connected to the auxiliary instance, not connected to the target, but with recovery catalog connection
- From backup, connected to the auxiliary instance, not connected to the target and the recovery catalog

You can duplicate databases by using the RMAN command line or with Enterprise Manager Cloud Control.

Duplicating an Active Database with “Push”



- “Push” method based on image copies
- Via Enterprise Manager or RMAN command line
- With network (use the FROM ACTIVE DATABASE clause)
- With a customized SPFILE

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

You can instruct the source database to send a “clone” of itself directly to the auxiliary instance by using Enterprise Manager or the FROM ACTIVE DATABASE clause of the RMAN DUPLICATE command.

Preexisting backups are neither needed nor used for this operation. The online image copies are created by the source database and directly transmitted via Oracle Net (they are not written to disk) when using the FROM ACTIVE DATABASE clause of the RMAN DUPLICATE command. The source database can be open or mounted.

RMAN connects as TARGET to the source database instance and as AUXILIARY to the auxiliary instance (as shown in the slide).

The required files (data files, control files, SPFILE, and archive log copies) are copied from the source to an auxiliary instance via an inter-instance network connection. RMAN then uses a “memory script” (one that is contained only in memory) to complete recovery and open the database.

This method of active database duplication is referred to as the “push”-based method.

Comparing the “Push” and “Pull” Methods of Duplication

- The push process is based on image copies.
- The pull process is based on backup sets.
 - RMAN uses the pull method when it finds:
 - USING BACKUPSET
 - SECTION SIZE
 - Encryption
 - Compression
 - The pull method requires connections to target and auxiliary instances.

```
RMAN> SET ENCRYPTION ...;
RMAN> DUPLICATE TARGET DATABASE TO orcl ldb
      FROM ACTIVE DATABASE
      [USING BACKUPSET]
      [SECTION SIZE ...]
      [USING COMPRESSED BACKUPSET] ...;
```

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

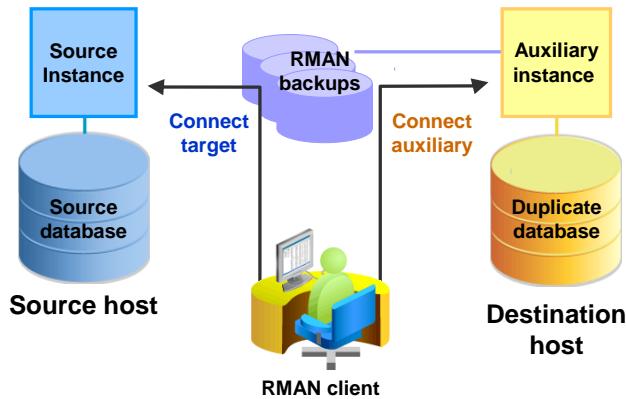
The “pull” (or restore) process: A connection is first established with the source database instance. The auxiliary instance then retrieves the required database files from the source database as backup sets. A restore operation is performed from the auxiliary instance. Therefore, fewer resources are used on the source database. Both TNS connections are required on target and auxiliary instances.

Based on the DUPLICATE clauses, RMAN dynamically determines which process to use (push or pull). This ensures that existing customized scripts continue to function.

- When you specify USING BACKUPSET, RMAN uses the pull method.
- When you specify SET ENCRYPTION before the DUPLICATE command, RMAN automatically uses the pull method and creates backup sets. The backups sent to the destination are encrypted.
- The SECTION SIZE clause divides data files into subsections that are restored in parallel across multiple channels on the auxiliary database. For an effective use of parallelization, allocate more AUXILIARY channels.
- With the USING COMPRESSED BACKUPSET clause, the files are transferred as compressed backup sets. RMAN uses unused block compression while creating backups, thus reducing the size of backups that are transported over the network.

Duplicating a Database with a Target Connection

- RMAN connects to the target (source database instance).
- RMAN connects to the auxiliary instance.



ORACLE®

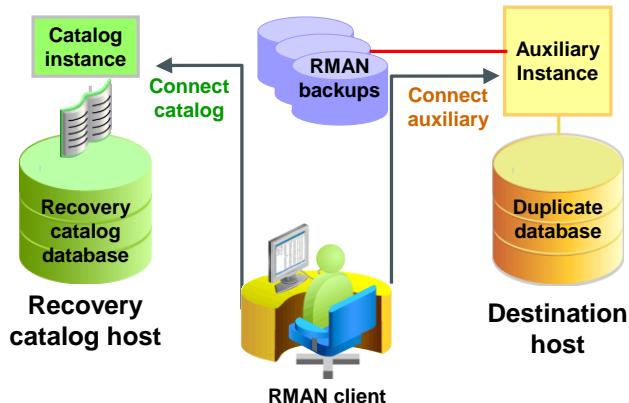
Copyright © 2020, Oracle and/or its affiliates.

When you duplicate a database with a target database connection, RMAN can obtain metadata about backups either from the target database control file or from the recovery catalog.

The diagram illustrates backup-based duplication with a target connection. RMAN connects to the source database instance and the auxiliary instance. Optionally, RMAN can connect to a recovery catalog database (not shown in the graphic). The destination host must have access to the RMAN backups required to create the duplicate database.

Duplicating a Database with a Recovery Catalog

- RMAN connects to a recovery catalog instance for backup metadata.
- RMAN connects to the auxiliary instance, which must have access to the RMAN backups.



ORACLE®

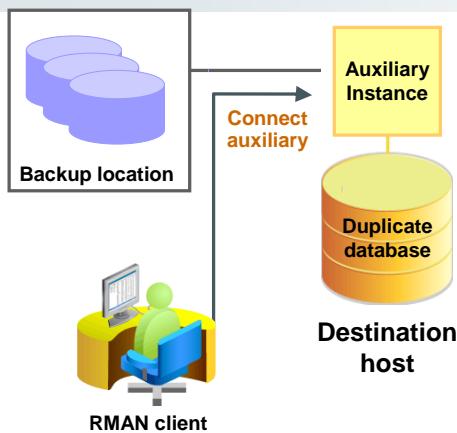
Copyright © 2020, Oracle and/or its affiliates.

When you duplicate a database without a target database connection, but with a recovery catalog, RMAN uses the recovery catalog to obtain metadata about the backups.

The diagram illustrates backup-based duplication without a target connection. RMAN connects to a recovery catalog database instance and the auxiliary instance. The destination host must have access to the RMAN backups required to create the duplicate database.

Duplicating a Database Without a Recovery Catalog or Target Connection

RMAN connects to the auxiliary instance, which must have access to a backup location.



ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

When you duplicate a database without a target database connection and recovery catalog, RMAN uses a backup location where all necessary backups and copies reside.

The diagram illustrates backup-based duplication without connections to the target or to the recovery catalog database instance. A disk backup location containing all the backups or copies for duplication must be available to the destination host.

Summary

In this lesson, you should have learned how to:

- List the purposes of creating a duplicate database
- Choose a technique for duplicating a database



Copyright © 2020, Oracle and/or its affiliates.

Creating a Backup-Based Duplicate Database

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

Objectives



After completing this lesson, you should be able to:

- Use an RMAN to create a backup-based duplicate database
- Describe the RMAN duplication operation
- Clone an active PDB into an existing CDB



Copyright © 2019, Oracle and/or its affiliates.

1 - 2

For Instructor Use Only.
This document should not be distributed.

Creating a Backup-Based Duplicate Database

1. Create an Oracle password file for the auxiliary instance.
2. Establish Oracle Net connectivity to the auxiliary instance.
3. Create an initialization parameter file for the auxiliary instance.
4. Start the auxiliary instance in NOMOUNT mode.
5. Mount or open the target database.
6. Ensure that backups and archived redo log files are available.
7. Allocate auxiliary channels if needed.
8. Execute the DUPLICATE command.



Copyright © 2020, Oracle and/or its affiliates.

It is important to understand these basic steps and the RMAN database duplication process.

If you are using the Enterprise Manager interface, wizards can perform most steps for you. If you are creating a duplicate database with the command-line interface, you need to perform the steps manually. You can also use the EM interface as a test or sample and use the output log as a basis for scripting your own database duplication.

The basic steps for creating a duplicate database are outlined in the slide. More details are provided in this lesson for some of the steps.

Creating an Initialization Parameter File for the Auxiliary Instance

Specify parameters as follows:

- **DB_NAME** (required)
 - If the duplicate database is in the same Oracle home as the target database, names must be different.
 - Use the same value in the `DUPLICATE` command.
- **CONTROL_FILES** (only required without the `SET NEWNAME` option and OMF)



Copyright © 2020, Oracle and/or its affiliates.

You must create a text initialization parameter file for the auxiliary instance. The text initialization parameter file must reside on the same host as the RMAN client that you use to execute the `DUPLICATE` command.

Take note of the requirements for each of the following parameters:

- **DB_NAME:** If the target database and the duplicate database are in the same Oracle home, you must set `DB_NAME` to a different name. If they are in different Oracle homes, you must ensure that the name of the duplicate database differs from the other names in its Oracle home. Be sure to use the same database name that you set for this parameter when you execute the `DUPLICATE` command.
- **CONTROL_FILES:** This parameter is required when you are *not* using the `SET NEWNAME` option and Oracle Managed Files (OMF).

Note: Be sure to verify the settings of all initialization parameters that specify path names. Verify that all specified paths are accessible on the duplicate database host.

Specifying New Names for Your Destination

Available techniques:

- SET NEWNAME command
- CONFIGURE AUXNAME command (deprecated for recovery set data files)
- DB_FILE_NAME_CONVERT parameter with the DUPLICATE command



Copyright © 2020, Oracle and/or its affiliates.

You can use the following techniques to specify new names for data files:

- Include the SET NEWNAME FOR DATAFILE command within a RUN block to specify new names for the data files.
- Use the CONFIGURE AUXNAME command.

CONFIGURE AUXNAME is an alternative to SET NEWNAME. The difference is that after you configure the auxiliary name the first time, additional DUPLICATE commands reuse the configured settings. In contrast, you must reissue the SET NEWNAME command every time you execute the DUPLICATE command.

Note: SET NEWNAME replaces CONFIGURE AUXNAME for recovery set data files.

- Specify the DB_FILE_NAME_CONVERT parameter with the DUPLICATE command.

Using the SET NEWNAME Clauses

- SET NEWNAME clauses enable you to specify a default name format for all files in a database or in a named tablespace.
- The default name is used for DUPLICATE, RESTORE, and SWITCH commands in the RUN block.
- It enables you to set file names with a single command rather than setting each file name individually.

```
SET NEWNAME FOR DATABASE  
TO {NEW|'formatSpec'};
```



Copyright © 2020, Oracle and/or its affiliates.

You can use SET NEWNAME to specify the default name format for all data files in a named tablespace and all data files in the database.

The order of precedence for the SET NEWNAME command is as follows:

1. SET NEWNAME FOR DATAFILE and SET NEWNAME FOR TEMPFILE
2. SET NEWNAME FOR TABLESPACE
3. SET NEWNAME FOR DATABASE

Example:

```
RUN  
{  
  SET NEWNAME FOR DATABASE TO '/u01/app/oracle/oradata/dupldb/%b';  
  DUPLICATE TARGET DATABASE TO dupldb  
  LOGFILE  
    GROUP 1 ('/u01/app/oracle/oradata/dupldb redo01a.log',  
             '/u01/app/oracle/oradata/dupldb redo01b.log') SIZE 50M REUSE,  
    GROUP 2 ('/u01/app/oracle/oradata/dupldb redo02a.log',  
             '/u01/app/oracle/oradata/dupldb redo02b.log') SIZE 50M REUSE,  
    GROUP 3 ('/u01/app/oracle/oradata/dupldb redo03a.log',  
             '/u01/app/oracle/oradata/dupldb redo03b.log') SIZE 50M REUSE;  
}
```

Substitution Variables for SET NEWNAME

Syntax Element	Description
%b	Specifies the file name without the directory path *NEW*
%f	Specifies the absolute file number of the data file for which the new name is generated
%I	Specifies the DBID
%N	Specifies the tablespace name
%U	Specifies a system-generated file name of the format: data-D-%d_id-%I_TS-%N_FNO-%f

```
RUN
{ SET NEWNAME FOR DATAFILE 1 TO '/oradata1/system01.dbf';
SET NEWNAME FOR DATAFILE 2 TO '/oradata2/sysaux01.dbf';
SET NEWNAME FOR DATAFILE 3 TO '/oradata3/undotbs01.dbf';
SET NEWNAME FOR DATAFILE 4 TO '/oradata4/users01.dbf';
SET NEWNAME FOR TABLESPACE example TO '/oradata5/%b';
DUPLICATE TARGET DATABASE TO dupldb; }
```



Copyright © 2020, Oracle and/or its affiliates.

To avoid possible name collisions when restoring to another location, use the substitution variables of the SET NEWNAME command. Specify at least one of the following substitution variables: %b, %f, and %U. %I and %N are optional variables.

The example shows the SET NEWNAME FOR TABLESPACE command to set default names with a substitution variable, together with explicit SET NEWNAME clauses.

Specifying Parameters for File Naming

Alternatively, specify the following parameters to explicitly control the naming of the files of your auxiliary database:

- **CONTROL_FILES**
- **DB_FILE_NAME_CONVERT**
- **LOG_FILE_NAME_CONVERT**

```
CONTROL_FILES='/u01/app/oracle/oradata/aux/control01.ctl',
              '/u01/app/oracle/oradata/aux/control02.ctl',
              '/u01/app/oracle/oradata/aux/control03.ctl'
DB_FILE_NAME_CONVERT='/u01/app/oracle/oradata/orcl',
                     '/u01/app/oracle/oradata/aux'
LOG_FILE_NAME_CONVERT='/u01/app/oracle/oradata/orcl',
                     '/u01/app/oracle/oradata/aux'
```



Copyright © 2020, Oracle and/or its affiliates.

RMAN generates names for the required database files when you execute the **DUPLICATE** command. You can control the naming of the files by specifying the following initialization parameters in the auxiliary instance initialization parameter file:

- **CONTROL_FILES**: Specify the names of the control files in this parameter. If you do not set the names via this parameter, the Oracle server creates an Oracle-managed control file in a default control destination. Refer to the **SQL CREATE CONTROLFILE** command in the **SQL Reference** manual for specific information.
- **DB_FILE_NAME_CONVERT**: This parameter is used to specify the names of data files for the auxiliary database. It has the format **DB_FILE_NAME_CONVERT = 'string1', 'string2'**, where *string1* is the pattern of the target database file name and *string2* is the pattern of the auxiliary database file name. You can also specify the **DB_FILE_NAME_CONVERT** parameter as an option to the **DUPLICATE DATABASE** command.
- **LOG_FILE_NAME_CONVERT**: This parameter is used to specify the names of the redo log files for the auxiliary database. It has the format **LOG_FILE_NAME_CONVERT = 'string1', 'string2'**, where *string1* is the pattern of the target database file name and *string2* is the pattern of the auxiliary database file name. You can also use the **LOGFILE** clause of the **DUPLICATE DATABASE** command to specify redo log file names.

As an alternative to using the initialization parameters to control the naming of the files, you can use the following techniques to rename the redo log files:

- Use the **LOGFILE** clause of the **DUPLICATE** command.
- Set the Oracle Managed Files initialization parameters: **DB_CREATE_FILE_DEST**, **DB_CREATE_ONLINE_DEST_n**, or **DB_RECOVERY_FILE_DEST**.

Starting the Instance in NOMOUNT Mode

- Start the auxiliary instance in NOMOUNT mode.
- RMAN can create an SPFILE.

```
SQL> startup nomount pfile='$HOME/auxinstance/initAUX.ora'
ORACLE instance started.

Total System Global Area  285212672 bytes
Fixed Size                  1218992 bytes
Variable Size                92276304 bytes
Database Buffers            188743680 bytes
Redo Buffers                 2973696 bytes

SQL> create spfile
  2  from pfile='$HOME/auxinstance/initAUX.ora';
                                         Not needed in course practice

File created.
```



Copyright © 2020, Oracle and/or its affiliates.

After you have created the text initialization parameter file, invoke SQL*Plus to start the auxiliary instance in NOMOUNT mode.

RMAN creates a default server parameter file for the auxiliary instance if the following conditions are true:

- Duplication does not involve a standby database.
- Server parameter files are not being duplicated.
- The auxiliary instance was not started with a server parameter file.

If these conditions are not met, create a server parameter file (SPFILE) from your text initialization parameter file. You can execute CREATE SPFILE before or after you have started the instance. See *Oracle Database SQL Language Reference* for details on the CREATE SPFILE command and examples.

Ensuring That Backups and Archived Redo Log Files Are Available

- Backups of all target database data files must be accessible on the duplicate host.
- Backups can be a combination of full and incremental backups.
- Archived redo log files needed to recover the duplicate database must be accessible on the duplicate host.
- Archived redo log files can be:
 - Backups on a media manager
 - Image copies
 - Actual archived redo log files



Copyright © 2020, Oracle and/or its affiliates.

The backups needed to restore the data files must be accessible on the duplicate host. The backup copies must be available on local storage or on a networked storage solution such as a network file system (NFS) mounted file system or a storage area network (SAN) LUN. You do not need a whole database backup. RMAN can use a combination of full and incremental backups of individual data files during the duplication process.

Archived redo logs required to recover the duplicate database to the desired point in time must also be accessible. The archived redo log files can be backups, image copies, or the actual archived redo logs. The backups or copies can be transferred to the local disk of the duplicate database node or mounted across a network by some means such as network file system (NFS).

Allocating Auxiliary Channels

- Auxiliary channels specify a connection between RMAN and an auxiliary database instance.
- If automatic channels are not configured, allocate auxiliary channels:
 - Start RMAN with a connection to the target database instance, the auxiliary instance, and recovery catalog if applicable.
 - Allocate at least one auxiliary channel within the `RUN` block.

```
$ rman target sys/oracle_4U@trgt auxiliary
      sys/oracle_4U@auxdb
RMAN> RUN
  {ALLOCATE AUXILIARY CHANNEL aux1 DEVICE TYPE DISK;
   ALLOCATE AUXILIARY CHANNEL aux2 DEVICE TYPE DISK;
   ...
DUPLICATE TARGET DATABASE to auxdb; . . .
```



Copyright © 2020, Oracle and/or its affiliates.

If you do not have automatic channels configured, manually allocate at least one auxiliary channel before issuing the `DUPLICATE` command. The `ALLOCATE AUXILIARY CHANNEL` command must be within the same `RUN` block as the `DUPLICATE` command.

The channel type specified on the `ALLOCATE AUXILIARY CHANNEL` command must match the media where the backups of the target database are located.

- If the backups reside on disk, you can allocate more than one channel to reduce the time it takes for the duplication process.
- For tape backups, you can specify the number of channels that correspond to the number of devices available.

The auxiliary instance must be started with the `NOMOUNT` option, and the target database must be mounted or open.

Understanding the RMAN Duplication Operation

When you execute the `DUPLICATE` command, RMAN performs the following operations:

- 1A. Creates a control file server parameter file for the auxiliary instance (for active and for backup-based duplication with target connection) *or*
- 1B. Restores from backup (for standby database and for backup-based duplication without target connection)
2. Mounts the backup control file
3. For backup-based duplication: Selects the backups for restoring the data files to the auxiliary instance
4. Restores the target data files to the duplicate database
5. Performs incomplete recovery using all available incremental backups and archived redo log files



Copyright © 2020, Oracle and/or its affiliates.

When you execute the `DUPLICATE` command, RMAN performs the operations listed in the slide.

- 1A. RMAN creates a default server parameter file for the auxiliary instance if the following conditions are true:
 - Duplication does not involve a standby database.
 - Server parameter files are not being duplicated.
 - The auxiliary instance was not started with a server parameter file.
- 1B. RMAN restores from backup—always for the standby database and for backup-based duplication without target connection.
2. RMAN mounts the restored or the copied backup control file from the active database.
3. For backup-based duplication: RMAN uses the RMAN repository to select the backups for restoring the data files to the auxiliary instance.
4. RMAN restores and copies the duplicate data files.
5. RMAN recovers the data files with incremental backups and archived redo log files to a noncurrent point in time. RMAN must perform database point-in-time recovery, even when no explicit point in time is provided for duplication. Point-in-time recovery is required because the online redo log files in the source database are not backed up and cannot be applied to the duplicate database. The farthest point of recovery of the duplicate database is the most recent redo log file archived by the source database.

Understanding the RMAN Duplication Operation

When you execute the `DUPLICATE` command, RMAN performs the following operations:

6. Shuts down and restarts the auxiliary instance in `NOMOUNT` mode
7. Creates a new control file, which then creates and stores the new `DBID` in the data files
8. Opens the duplicate database with the `RESETLOGS` option
9. Creates the online redo log files for the duplicate database

Note: The database duplication process attempts to resume from the point-of-failure upon re-execution.



Copyright © 2020, Oracle and/or its affiliates.

6. RMAN shuts down and restarts the database instance in `NOMOUNT` mode.
7. RMAN creates a new control file, which then creates and stores the new, unique database identifier `DBID` in the data files of the duplicated database.
8. RMAN opens the duplicate database with the `RESETLOGS` option.
9. RMAN creates the online redo log files for the duplicate database.

Note: If the `DUPLICATE DATABASE` command fails, you can re-execute the `DUPLICATE DATABASE` command and the duplication process attempts to resume from the point of failure.

Specifying Options for the DUPLICATE Command

You can specify the following options with the DUPLICATE command:

Option	Purpose
SKIP READONLY	Excludes read-only tablespaces
SKIP TABLESPACE	Excludes named tablespaces
TABLESPACE	Includes named tablespaces
NOFILENAMECHECK	Prevents checking of file names
OPEN RESTRICTED	Enables RESTRICTED SESSION automatically
NOOPEN	Leaves duplicate database in MOUNT mode



Copyright © 2020, Oracle and/or its affiliates.

Specify additional options when executing the DUPLICATE command as appropriate.

- **SKIP READONLY:** Use to exclude read-only tablespace data files.
- **SKIP TABLESPACE:** Use to exclude tablespaces from the target database. You cannot exclude the SYSTEM tablespace or tablespaces containing undo or rollback segments.
- **TABLESPACE:** Use to include tablespaces from the target database.
- **NOFILENAMECHECK:** Use to prevent RMAN from checking whether target database data files with the same name as duplicate database data files are in use. You must specify this option when the target database and duplicate database data files and redo log files use the same names. You would typically use this when you create a duplicate database on a host that has the same disk configuration, directory structure, and file names as the target database host. If you do not specify NOFILENAMECHECK in this situation, RMAN returns an error.
- **OPEN RESTRICTED:** Use to enable RESTRICTED SESSION automatically after the database is opened.
- **NOOPEN:** Use to finish with the duplicate database in MOUNT mode. Use this option before:
 - Modifying block change tracking
 - Configuring fast incremental backups or flashback database settings
 - Moving the location of the database (for example, to ASM)
 - Upgrading a database

Using Additional DUPLICATE Command Options

Option	Purpose
NOREDO	Signals RMAN that the application of redo logs should be suppressed during recovery Must be used with targetless DUPLICATE when target database is in NOARCHIVELOG mode at backup time Can also be used to explicitly state that no archived redo log files should be applied
UNDO TABLESPACE	Must be specified when target database is not open and there is no recovery catalog connection so that RMAN does not check the tablespace for SYS-owned objects



Copyright © 2020, Oracle and/or its affiliates.

- **NOREDO:** The NOREDO option is used to signal RMAN that redo logs should not be applied during the recovery phase of the duplication operation. This option should be specified when the database was in NOARCHIVELOG mode at the time of the backup or when the archived redo log files are not available for use during the duplication operation. This option is appropriate if a database that is currently in ARCHIVELOG mode is being duplicated to a point in time when it was in NOARCHIVELOG mode.
If you are planning a targetless DUPLICATE operation and the database is in NOARCHIVELOG mode, you must use the NOREDO option to inform RMAN of the database mode. Without a connection to the target database, RMAN cannot determine the mode.
- **UNDO TABLESPACE:** RMAN checks that there are no objects belonging to the SYS user in any of the duplicated tablespaces during nonwhole database duplication. The SYSTEM, SYSAUX, and undo segment tablespaces are excluded from this check. However, if the target database is not open and a recovery catalog is not being used during the duplication, RMAN cannot obtain the undo tablespace names. So you must use the UNDO TABLESPACE option to provide the names of undo segment tablespaces.

Duplicating Selected PDBs in a CDB

- A single pluggable database:

```
RMAN> DUPLICATE DATABASE TO cdb1 PLUGGABLE DATABASE pdb1;
```

- Several pluggable databases:

```
RMAN> DUPLICATE DATABASE TO cdb1 PLUGGABLE DATABASE pdb1, pdb3;
```

- All pluggable databases except one:

```
RMAN> DUPLICATE DATABASE TO cdb1 SKIP PLUGGABLE DATABASE pdb3;
```

- A PDB and tablespaces of other PDBs:

```
RMAN> DUPLICATE DATABASE TO cdb1  
      PLUGGABLE DATABASE pdb1 TABLESPACE pdb2:users;
```



Copyright © 2020, Oracle and/or its affiliates.

RMAN enables you to duplicate multitenant container databases including all or individual PDBs by using the `DUPLICATE` command.

To duplicate PDBs, you must create the auxiliary instance as a CDB. To do so, start the instance with the `ENABLE_PLUGGABLE_DATABASE=TRUE` in the initialization parameter file. When you duplicate one or more PDBs, RMAN also duplicates the CDB root (`CDB$ROOT`) and the PDB seed (`PDB$SEED`). The resulting duplicate database is a fully new, functional CDB that contains the CDB root, the CDB seed, and the duplicated PDBs.

The first example shows how to duplicate a single PDB, the second one how to duplicate a set of PDBs, the third one how to duplicate all the databases in the CDB, except a PDB, and the last one how to duplicate a set of tablespaces within a PDB.

You must be logged in to the CDB root as a user who is granted the `SYSDBA` or `SYSBACKUP` role.

Cloning an Active PDB into an Existing CDB

- Duplicate a PDB or PDB tablespaces in active mode to an existing opened CDB.
- Clone only one PDB at a time.
- Set the COMPATIBLE initialization parameter to 18.1 or higher.
- Set the destination CDB in READ WRITE mode.
- Set the REMOTE_RECOVERY_FILE_DEST initialization parameter in the destination CDB to the location where to restore foreign archive log files.

```
RMAN> DUPLICATE PLUGGABLE DATABASE pdb1 AS pdb2 FROM ACTIVE DATABASE  
      DB_FILE_NAME_CONVERT ('cdb1', 'cdb2');
```



Copyright © 2020, Oracle and/or its affiliates.

An active PDB can be duplicated directly into an open CDB.

The passwords for target and auxiliary connections must be the same when using active duplicate.

In the auxiliary instance, define the location where to restore the foreign archive log files via the new initialization parameter, REMOTE_RECOVERY_FILE_DEST.

RMAN should be connected to the CDB root of the target and auxiliary instances.

Limitations:

- Non-CDB to PDB duplication is not supported.
- Encryption is not supported for PDB cloning.
- SPFILE, NO STANDBY, FARSYNC STANDBY, and LOG_FILE_NAME_CONVERT keywords are not supported.

NORESUME, DB_FILE_NAME_CONVERT, SECTION SIZE, and USING COMPRESSED BACKUPSET keywords are supported.

Example: Duplicating PDB1 from CDB1 to CDB2 as PDB1

1. Set the `REMOTE_RECOVERY_FILE_DEST` initialization parameter in CDB2.

```
SQL> ALTER SYSTEM SET REMOTE_RECOVERY_FILE_DEST='/dir_to_restore_archive_log_files';
```

2. Connect to the source (TARGET for DUPLICATE command): CDB1

3. Connect to the existing CDB2 that acts as the auxiliary instance:

```
RMAN> CONNECT TARGET "sys/oracle_4U@cdb1 AS SYSDBA"  
RMAN> CONNECT AUXILIARY "sys/oracle_4U@cdb2 AS SYSDBA"
```



4. Start the duplication.

```
RMAN> DUPLICATE PLUGGABLE DATABASE pdb1 TO cdb2 FROM ACTIVE DATABASE;
```

ORACLE

Copyright © 2020, Oracle and/or its affiliates.

The example shows a duplication of PDB1 from CDB1 to the existing CDB2 as PDB1.

To perform this operation, connections to the source (TARGET) CDB1 and to the destination (AUXILIARY) CDB2 are required.

The restore location for the foreign archive log files in the auxiliary instance is defined by the `REMOTE_RECOVERY_FILE_DEST` initialization parameter.

The `DUPLICATE` command specifies that the operation is performed while the source PDB1 is opened.

CDB2 needs to be opened in read/write mode.

Example: Duplicating PDB1 from CDB1 to CDB2 as PDB2

1. Set the `REMOTE_RECOVERY_FILE_DEST` initialization parameter in CDB2.

```
SQL> ALTER SYSTEM SET REMOTE_RECOVERY_FILE_DEST='/dir_to_restore_archive_log_files';
```

2. Connect to the source (TARGET for DUPLICATE command): CDB1

3. Connect to the existing CDB2 that acts as the auxiliary instance:

```
rman TARGET sys@cdb1 AUXILIARY sys@cdb2
```



4. Start the duplication.

```
RMAN> DUPLICATE PLUGGABLE DATABASE pdb1 AS pdb2 TO cdb2 FROM ACTIVE DATABASE;
```

ORACLE®

Copyright © 2020, Oracle and/or its affiliates.

The example shows a duplication of PDB1 from CDB1 to the existing CDB2 as PDB2.

To perform this operation, connections to the source (TARGET) CDB1 and to the destination (AUXILIARY) CDB2 are required.

The restore location for the foreign archive log files in the auxiliary instance is defined by the `REMOTE_RECOVERY_FILE_DEST` initialization parameter.

The `DUPLICATE` command specifies that the operation is performed while the source PDB1 is opened.

CDB2 needs to be opened in read/write mode.

Summary

In this lesson, you should have learned how to:

- Use an RMAN to create a backup-based duplicate database
- Describe the RMAN duplication operation
- Clone an active PDB into an existing CDB



Practice Overview

- Duplicating a database
- Duplicating a PDB into an existing CDB



Copyright © 2020, Oracle and/or its affiliates.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.