Oracle8™

Backup and Recovery Guide

Release 8.0

December, 1997

Part No. A58396-01

ORACLE®

Enabling the Information Age™

Oracle8 Backup and Recovery Guide

Part No. A58396-01

Release 8.0

Copyright © 1997, Oracle Corporation. All rights reserved.

Primary Authors: Connie Dialeris, Joyce Fee

Contributors: Bill Bridge, Sandra Cheevers, John Frazzini, Tim Berry-Hart, Gordon Larimer, Bill Lee, Diana Lorentz, Greg Pongracz, Lyn Pratt, Tuomas Pystynen, Daniel Semler, Slartibartfast, Steve Wertheimer, Min Zhou

Graphic Designer: Valerie Moore

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Programs.

This Program contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright patent and other intellectual property law. Reverse engineering of the software is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free.

If this Program is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend Programs delivered subject to the DOD FAR Supplement are 'commercial computer software' and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are 'restricted computer software' and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52..227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle, Net8, and SQL*Plus are registered trademarks of Oracle Corporation. Oracle8, Server Manager, Enterprise Manager, Recovery Manager, Oracle Parallel Server and PL/SQL are trademarks of Oracle Corporation.

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

Contents

Send Us Your Comments xiii

Preface xv

1 Why Perform Backups?

What Is a Backup? 1-2
- Why Are Backups Important? 1-2
- When to Take Backups 1-2
- Types of Failures 1-3

Physical Database Structures 1-4
- Control Files 1-4
- Online Redo Log Files 1-5
- Datafiles 1-5
- Rollback Segments 1-6
- Archived Logs 1-7

2 What Are You Backing Up?

The Online Redo Log 2-2
- Online Redo Log File Contents 2-2
- How Online Redo Log Files Are Written 2-3
- Checkpoints 2-5
- Multiplexed Online Redo Log Files 2-9
- Threads of Online Redo Log and the Oracle Parallel Server 2-13

The Archived Redo Log 2-13

The Mechanics of Archiving 2-14
Archived Redo Log File Contents 2-15
Duplexing the Archived Redo Log 2-15
Database Archiving Modes 2-16
NOARCHIVELOG Mode (Media Recovery Disabled) 2-16
ARCHIVELOG Mode (Media Recovery Enabled) 2-16
Control Files 2-19
Control File Contents 2-19
Multiplexed Control Files 2-20
Types of Backups 2-21
Whole Database Backups 2-21
Tablespace Backups 2-24
Datafile Backups 2-24
Control File Backups 2-24
Archivelog Backups 2-25
Backup Formats 2-26
Backup Sets 2-26
Datafile Copies 2-27
Operating System Backups 2-27
Logical Backups 2-28

3 When to Perform Backups

Guidelines for Database Backups 3-2
Perform Backups Frequently and Regularly 3-2
Backup Appropriate Portions of the Database When Making Structural Changes 3-3
Back Up Often-used Tablespaces Frequently 3-3
Back Up after Performing Unrecoverable / Unlogged Operations 3-4
Keep Older Backups 3-4
Database Backups After Using the RESETLOGS Option 3-5
Export Database Data for Added Protection and Flexibility 3-6
Consider Distributed Database Backups 3-6
Test Backup and Recovery Strategies 3-7
Creating a Backup Strategy 3-7
Backup Strategies in NOARCHIVELOG Mode 3-7
Backup Strategies in ARCHIVELOG Mode 3-8

Backing Up Online Redo Logs 3-9
The Dangers Associated with Backing Up Online Redo Logs 3-12

4 Choosing Recovery Strategies

Recovery Concepts and Strategies 4-2
Important Recovery Data Structures 4-3
Recovery Planning 4-5
Factors Determining Your Recovery Strategy 4-5
Recovery Operations 4-7

5 Choosing a Backup Method

Backup Methods and Requirements 5-2
Recovery Manager 5-2
Operating System (O/S) 5-2
Export 5-3
Enterprise Backup Utility 5-3
Recovery Manager 5-3
Using Oracle Enterprise Manager to Perform Recovery Manager Backups 5-3
Recovery Manager is Different from Traditional Operating System Backups 5-4
Backups to Disk 5-4
Backups to Sequential Media 5-5
Feature Comparison of Backup Methods 5-7

6 Getting Started with Recovery Manager

Decisions to Make Before Using Recovery Manager 6-2
Will You Use a Recovery Catalog? 6-2
Decide Whether or Not to Use Password Files 6-6
Decide How to Back Up init.ora Files and Password Files 6-7
Recovery Manager Connection Options 6-7
Connecting to Recovery Manager Without a Recovery Catalog 6-7
Connecting to Recovery Manager With a Recovery Catalog 6-8
Running Recovery Manager Commands 6-9
Running Recovery Manager Commands Interactively: Example 1 6-9
Using Command Files: Example 2 6-10

Using Stored Scripts: Example 3 6-10
Specifying Time Parameters in Recovery Manager 6-11
Recovery Manager Sample Scripts and Scenarios 6-11
Prerequisites for Performing Backups to Tape 6-11
Linking with a Media Manager 6-12
Generating Unique File Names 6-12
Know Your Media Manager's Maximum File Size Limit 6-13

7 Recovery Manager Concepts

Introduction to Recovery Manager 7-2
Backing Up to Sequential Media 7-4
The Recovery Catalog 7-5
Operating with a Recovery Catalog 7-5
Operating without a Recovery Catalog 7-7
Snapshot Control File 7-9
Stored Scripts 7-9
Recovery Manager Backup Types 7-10
Backup Sets 7-10
Full and Incremental Backup Sets 7-12
Image Copies 7-17
Corruption Detection 7-18
Channel Control 7-19
Parallelization 7-20
Factors Affecting Degree of Parallelization 7-20
Multiplexed Backup Sets 7-22
Report Generation 7-23
User Tags for Recovery Manager Backups 7-24
Backup Constraints 7-25
Restore Constraints 7-26
Integrity Checking 7-26
Fractured Block Detection During Open Database Backups in Recovery Manager 7-26
Tracking Archive Logs 7-27
Cataloging Image Copies and Archive Logs 7-27

8 Performing Backup and Recovery with Recovery Manager

Installing the Recovery Catalog 8-2
Registering a Database 8-2
Maintaining the Recovery Catalog 8-3
Registering a Target Database with the Recovery Catalog 8-3
Resetting the Information in the Recovery Catalog 8-4
Resynchronizing the Recovery Catalog with a Target Database 8-4
Changing the Availability of a Backup Set or File Copy 8-6
Cataloging User-Created Backup Files 8-8
Recovering a Lost or Damaged Recovery Catalog Database 8-9
Using Channel Control Commands 8-9
Channel Control Commands 8-10
Generating Reports 8-12
Generating a Report 8-13
Generating Lists 8-17
Maintaining Scripts 8-19
Creating and Replacing Scripts 8-19
Deleting Scripts 8-19
Printing Scripts 8-19
Configuring the Snapshot Control File Location 8-19
Backing Up Files 8-20
Performing Backups 8-21
Backing Up: Scenario 8-27
Copying Files 8-28
Copy Command Specifiers 8-28
Restoring Files 8-30
Database Point-In-Time Recovery 8-30
File Selection 8-30
Restore Destination for Datafiles 8-31
Restore Destination for Control Files 8-31
Replicating Control Files 8-31
Restore Destination for Archived Logs 8-32
Guidelines for Restoring Datafiles 8-32
Restore Command Operand List 8-33
Switching Datafiles 8-35

Recovering Datafiles 8-35
Guidelines for Recovering Datafiles 8-36
Database Point-In-Time Recovery 8-36
Recovery Commands 8-37
Recover Command Object List 8-38
Monitoring Backups and Restores 8-39
Connecting a Session to Channels 8-39
Monitoring Progress 8-40

9 Recovery Manager Scenarios

Backing Up in NOARCHIVELOG Mode 9-2
Backing Up Databases and Tablespaces 9-2
Backing Up a Database 9-2
Backing Up a Tablespace 9-3
Backing Up Individual Datafiles 9-3
Backing Up the Control File 9-3
Backing Up Archived Logs 9-3
Backing Up in a Parallel Server Environment 9-5
Copying Datafiles 9-5
Incremental Backups 9-6
Handling Errors 9-6
Using O/S Utilities To Make Copies 9-7
Keeping Backups 9-7
Restoring and Recovering 9-8
Restore and Recover When the Database Is Open 9-8
Restore 9-10
Database Point-In-Time Recovery 9-10
Querying the Recovery Catalog 9-11
Using the Report Command for Complex Recovery Catalog Queries 9-12

10 Recovery Manager Tablespace Point-in-Time Recovery

Introduction to Recovery Manager Tablespace Point-in-Time Recovery 10-2
Planning for Recovery Manager Tablespace Point-in-Time Recovery 10-3
Limitations 10-4
Recovery Manager TSPITR Planning Requirements 10-7

Performing Recovery Manager Tablespace Point-In-Time Recovery 10-9
Back Up Tablespaces After Recovery Manager TSPITR Is Complete 10-10
Tuning Considerations 10-10
Specify a New Name for Datafiles in Auxiliary Set Tablespaces 10-10
Set the Clone Name and Use a Datafile Copy for Recovery Manager TSPITR 10-11
Use the Converted Datafile Name 10-13

11 Performing Operating System Backups

Performing Backups 11-2
Listing Database Files Before Backup 11-2
Performing Whole Database Backups 11-3
Performing Tablespace, Datafile, Control File or Archivelog Backups 11-5
Performing Control File Backups 11-10
Recovering From a Failed Online Tablespace Backup 11-12
Using the Export and Import Utilities for Supplemental Database Protection 11-13
Using Export 11-13
Using Import 11-14

12 Recovering a Database

Coordinate Distributed Recovery 12-2
Coordinate Time-Based and Change-Based Distributed Database Recovery 12-2
Recover Database with Snapshots 12-3
Recovery Scenarios 12-3
Recovering a Closed Database 12-3
Recovering an Offline Tablespace in an Open Database 12-4
Starting Recovery During Instance Startup 12-4
Applying Redo Log Files 12-4
Applying Log Files 12-5
Interrupting Media Recovery 12-9
Restoring a Whole Database Backup, NOARCHIVELOG Mode 12-9
Specifying Parallel Recovery 12-11
Preparing for Media Recovery 12-11
Media Recovery Commands 12-11
Issues Common to All Media Recovery Operations 12-12
Performing Complete Media Recovery 12-15

Performing Closed Database Recovery 12-15
Performing Open-Database, Offline-Tablespace Individual Recovery 12-17
Performing Open-Database, Offline-Tablespace Individual Recovery 12-19
Performing Incomplete Media Recovery 12-21
Performing Cancel-based Recovery 12-22
Performing Time-based Recovery 12-26
Performing Change-based Recovery 12-31
Preparing for Disaster Recovery 12-35
Planning and Creating a Standby Database 12-35
Altering the Physical Structure of the Primary Database 12-39
Unrecoverable Objects and Recovery 12-43
Read-only Tablespaces and Recovery 12-44
Using a Backup Control File 12-44
Re-creating a Control File 12-44
Recovery Procedure Examples 12-45
Types of Media Failures 12-45
Loss of Datafiles 12-45
Loss of Online Redo Log Files 12-46
Loss of Archived Redo Log Files 12-51
Loss of Control Files 12-51
Recovery From User Errors 12-53

13 Performing Tablespace Point-in-Time Recovery

Introduction to Tablespace Point-in-Time Recovery 13-2
Planning for Tablespace Point-in-Time Recovery 13-3
Limitations Advisory 13-4
TSPITR Requirements 13-6
Performing Tablespace Point-In-Time Recovery 13-7
Step 1: Find Out if Objects Will be Lost when Performing TSPITR 13-8
Step 2: Research and Resolve Dependencies on the Primary Database 13-8
Step 3: Prepare the Primary Database for TSPITR 13-12
Step 4: Prepare the Parameter Files for the Clone Database 13-12
Step 5: Prepare Clone Database for TSPITR 13-13
Step 6: Recover the Clone Database 13-14
Step 7: Open the Clone Database 13-15

Step 8: Prepare the Clone Database for Export 13-15
Step 9: Export the Clone Database 13-15
Step 10: Copy the Recovery Set Clone Files to the Primary Database 13-15
Step 11: Import into the Primary Database 13-15
Step 12: Prepare the Primary Database for Use 13-16
Step 13: Back Up the Recovered Tablespaces in the Primary Database 13-16
Performing Partial TSPITR of Partitioned Tables 13-16
Step 1: Create a Table on the Primary Database for Each Partition Being Recovered 13-17
Step 2: Drop the Indexes on the Partition Being Recovered 13-17
Step 3: Exchange Partitions with Stand-Alone Tables 13-18
Step 4:Take the Recovery Set Tablespace Offline 13-18
Step 5: Create Tables at Clone Database 13-18
Step 6: Drop Indexes on Partitions Being Recovered 13-18
Step 7: Exchange Partitions with Stand-Alone Tables 13-18
Step 8: Export the Clone Database 13-18
Step 9: Copy the Recovery Set Datafiles to the Primary Database 13-19
Step 10: Import into the Primary Database 13-19
Step 11: Bring Recovery Set Tablespace Online 13-19
Step 12: Exchange Partitions with Stand-Alone Tables 13-19
Step 13: Back Up the Recovered Tablespaces in the Primary Database 13-20
Performing TSPITR of Partitioned Tables When a Partition Has Been Dropped 13-21
Step 1: Find the Low and High Range of the Partition that Was Dropped 13-21
Step 2: Create a Temporary Table 13-22
Step 3: Delete Records From Partitioned Table 13-22
Step 4: Take Recovery Set Tablespaces Offline 13-22
Step 5: Create Tables at Clone Database 13-22
Step 6: Drop Indexes on Partitions Being Recovered 13-22
Step 7: Exchange Partitions with Stand-Alone Tables 13-22
Step 8: Export the Clone Database 13-22
Step 9: Copy the Recovery Set Datafiles to the Primary Database 13-23
Step 10: Import into the Primary Database 13-23
Step 11: Bring Recovery Set Tablespace Online 13-23
Step 12: Insert Stand-Alone Tables into Partitioned Tables 13-23
Step 13: Back Up the Recovered Tablespaces in the Primary Database 13-24

Performing TSPITR of Partitioned Tables When a Partition Has Been Split 13-25
Step 1: Drop the Lower of the Two Partitions at the Primary Database 13-25
Step 2: Drop Indexes of Partitions Being Recovered 13-26
Step 3: Exchange Partitions with Stand-Alone Tables 13-26
Step 4: Take Recovery Set Tablespaces Offline 13-26
Step 5: Create Tables at Clone Database 13-26
Step 6: Drop Indexes in Partitions Being Recovered 13-26
Step 7: Exchange Partitions with Stand-Alone Tables 13-26
Step 8: Export the Clone Database 13-27
Step 9: Copy the Recovery Set Datafiles to the Primary Database 13-27
Step 10: Import into the Primary Database 13-27
Step 11: Bring Recovery Set Tablespace Online 13-27
Step 12: Exchange Partitions with Stand-Alone Tables 13-27
Step 13: Back Up the Recovered Tablespaces in the Primary Database 13-28
TSPITR Tuning Considerations 13-28
Recovery Set Location Considerations 13-28
Backup Control File Considerations 13-29

A Recovery Manager Command Syntax

allocateForDelete A-2
allocate A-3
archivelogRecordSpecifier A-4
atClause A-5
backup A-6
backupSpec A-7
catalog A-9
change A-10
connect A-11
copy A-12
copyOption A-13
createScript A-14
deleteScript A-15
host A-16
inputfile A-17
list A-18

listObjList A-19
needBackupOperand A-20
primary_key A-21
printScript A-22
recover A-23
register A-24
release A-25
replaceScript A-26
replicate A-27
report A-28
reportObject A-29
reset A-30
restoreObject A-31
restore A-32
restoreSpecOperand A-33
resync A-34
releaseForDelete A-35
rmanCmd A-36
reportObsoleteOperand A-37
run A-38
set A-39
setPragma A-40
sql A-41
switch A-42
untilClause A-43
validate A-44

Glossary

Index

Send Us Your Comments

Oracle8 Backup and Recovery Guide, Release 8.0

Part No. A58396-01

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

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

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

- email: infodev@us.oracle.com
- fax: (650) 506-7228 Attn: Server Technologies Documentation Manager
- letter: Server Technologies Documentation Manager
  Oracle Corporation
  500 Oracle Parkway
  Redwood Shores, CA 94065

If you would like a reply, please give your name, address, and telephone number below.

Preface

Welcome to the world of Oracle backup and recovery! This new book in the Oracle Server documentation library includes all of the conceptual and task-oriented information you will need to perform backup, restore, and recovery procedures whether you use the new Recovery Manager utility or existing operating system backups.

Attention: The *Oracle8 Backup and Recovery Guide* contains information that describes the features and functionality of the Oracle8 and the Oracle8 Enterprise Edition products. Oracle8 and Oracle8 Enterprise Edition have the same basic features. However, several advanced features are available only with the Enterprise Edition, and some of these are optional. For example, to perform automated tablespace point-in-time recovery (using Recovery Manager), you must have the Enterprise Edition.

For information about the differences between Oracle8 and the Oracle8 Enterprise Edition and the features and options that are available to you, please refer to *Getting to Know Oracle8 and the Oracle8 Enterprise Edition*.

Structure

This book contains the following parts and chapters.

Part / Chapter	Contents
PART 1	**Getting Started**
Chapter 1: Why Perform Backups?	Describes the fundamental data structures you need to understand before moving on to the following chapters. The structures described here are the fundamental building blocks for Oracle backup, restore and recovery procedures.
Chapter 2: What Are You Backing Up?	Describes in detail all of the fundamental concepts necessary to understand Oracle backup and recovery. These concepts also apply to the Recovery Manager utility.
Chapter 3: When to Perform Backups	A comprehensive description of the guidelines and strategies to follow for performing backups.
Chapter 4: Choosing Recovery Strategies	An overview of the guidelines and strategies to follow when performing database recovery. Here you will find descriptions of recovery situations along with recommendations for appropriate recovery procedures.
Chapter 5: Choosing a Backup Method	Describes different ways to implement the guidelines and strategies described in Chapter 3.
PART 2	**Backup and Recovery—Recovery Manager**
Chapter 6: Getting Started with Recovery Manager	Describes issues to consider before using Recovery Manager, as well as some tips for getting a “quick start” with Recovery Manager
Chapter 7: Recovery Manager Concepts	A comprehensive description of the Recover Manager utility, and includes fundamental Recovery Manager concepts.
Chapter 8: Performing Backup and Recovery with Recovery Manager	Provides instructions for using the Oracle Recovery Manager utility to manage your enterprise’s backup, restore and recovery operations.
Chapter 9: Recovery Manager Scenarios	Provides examples of specific situations in which you would use Recovery Manager and also includes appropriate Recovery Manager commands
Chapter 10: Recovery Manager Tablespace Point-in-Time Recovery	Describes all the planning and limitations to consider before using the step-by-step directions for using Recovery Manager to perform automated tablespace point-in-time recovery.
PART 3	**Backup and Recovery—Operating System**
Chapter 11: Performing Operating System Backups	Provides step-by-step instructions for performing operating system backups. The information in this appendix previously appeared in the *Oracle7 Server Administrator’s Guide*.

Part / Chapter	Contents
Chapter 12: Performing Operating System Recovery	Provides step-by-step instructions for performing operating system recovery. The information in this appendix previously appeared in the *Oracle7 Server Administrator's Guide*.
Chapter 13: Performing Tablespace Point-in-Time Recovery	Provides planning guidelines and step-by-step instructions for manually performing tablespace point-in-time recovery.
Appendix A, Recovery Manager Command Syntax	Includes the basic Recovery Manager command syntax diagrams.

Audience

This guide is for people who administer the backup, restore, and recovery operations of an Oracle database system.

Knowledge Assumed of the Reader

Readers of this guide are assumed to be familiar with relational database concepts. They are also assumed to be familiar with the operating system environment under which they are running Oracle.

How to Use This Guide

Every reader of this guide should read Chapter 1 of the *Oracle8 Concepts* manual, "Introduction to the Oracle Server." This overview of the concepts and terminology related to Oracle provides a foundation for the more detailed information in this guide. The rest of the *Oracle8 Concepts* manual explains the Oracle architecture and features, and how they operate in more detail.

Part I

Getting Started

1

Why Perform Backups?

This chapter introduces database concepts that are fundamental to backup and recovery, and explains why taking backups is crucial for successful database operations. The following topics are included:

- What Is a Backup?
- Physical Database Structures

What Is a Backup?

Simply speaking, a database backup is a representative copy of data. When the original data is lost, you can use the backup to reconstruct lost information (the physical files that constitute your Oracle database). This copy includes important parts of your database, such as the control file, archive logs and datafiles—structures described later in this chapter and throughout this book. In the event of a media failure, your database backup is the key to successfully recovering your data.

Why Are Backups Important?

Imagine the magnitude of lost revenue (not to mention the degree of customer dissatisfaction!) if the production database of a catalog company, express delivery service, bank or airline suddenly became unavailable, even for just 5 or 10 minutes; or, if you lose datafiles due to media failure and cannot restore or recover them because you do not have a backup. From your enterprise's perspective, the results may be quite grim. You must restore and recover your data quickly to resume operations. The key to your success in this situation is a well-defined backup and recovery strategy.

When to Take Backups

You should tailor your backup strategy to the needs of your business. For example, if it is acceptable to lose data in the event of a disk failure, you may not need to perform frequent backups. What if your database must be available twenty-four hours a day, seven days a week? In this case, your database would have to be frequently backed up. The frequency of your backups and types of backups performed is determined in large part by the needs of your business.

Types of Failures

An unfortunate aspect of every database system is the possibility of a system or hardware failure. The most common types of failure are described below.

statement and process failure	Statement failure occurs when there is a logical failure in the handling of a statement in an Oracle program (for example, the statement is not a valid SQL construction). When statement failure occurs, the effects (if any) of the statement are automatically undone by Oracle and control is returned to the user. A *process failure* is a failure in a user process accessing Oracle, such as an abnormal disconnection or process termination. The failed user process cannot continue work, although Oracle and other user processes can.
instance failure	*Instance failure* occurs when a problem arises that prevents an instance (system global area and background processes) from continuing work. Instance failure may result from a hardware problem such as a power outage, or a software problem such as an operating system crash. When an instance failure occurs, the data in the buffers of the system global area is not written to the datafiles. Oracle automatically recovers from instance failure when the database opens.
user or application error	User errors can require a database to be recovered to a point in time before the error occurred. For example, a user might accidentally delete data from a table that is still required (for example, payroll taxes). To allow recovery from user errors and accommodate other unique recovery requirements, Oracle provides for exact point-in-time recovery. For example, if a user accidentally deletes data, the database can be recovered to the instant in time before the data was deleted.

media (disk) failure	An error can arise when trying to write or read a file that is required to operate the database. This is called disk failure because there is a physical problem reading or writing physical files on disk. A common example is a disk head crash, which causes the loss of all files on a disk drive. Different files may be affected by this type of disk failure, including the datafiles, the redo log files, and the control files. Also, because the database instance cannot continue to function properly, the data in the database buffers of the system global area cannot be permanently written to the datafiles.

Physical Database Structures

The following sections give a brief overview of the physical database structures of an Oracle database. These topics will be covered in more detail in subsequent chapters of this book.

Control Files

Every Oracle database has a *control file*. A control file is an extremely important datafile that contains entries specifying the physical structure of the database, and provide database consistency information used during recovery. For example, it contains the following types of information:

- database name
- names and locations of a database's datafiles and redo log files
- time stamp of database creation
- backup information (when using the Recovery Manager utility)

Like the redo log, Oracle allows the control file to be mirrored for protection of the control file.

Use of Control Files

Every time an instance of an Oracle database is mounted, its control file is used to identify the datafiles and redo log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, a new datafile or redo log file is created), the database's control file is automatically modified by Oracle to reflect the change.

You should back up the control file any time there are structural changes to the database.

Online Redo Log Files

Every Oracle database has a set of two or more *redo log files*. The set of redo log files for a database is collectively known as the database's *redo log*. Oracle uses the redo log to record all changes made to data. For example, a failure has prevented modified data from being permanently written to the datafiles. In this situation, you can obtain the modified data from the redo log and permanently write it to the datafiles, all-the-while preventing loss of work.

Redo log files are critical in protecting a database against failures. To protect against a failure involving the redo log itself, Oracle allows the redo log to be *multiplexed*. This means Oracle will maintain two or more copies of the redo log on different disks.

You do not need to back up the online redo log, nor should you ever need to restore it.

The Use of Redo Log Files

The information in a redo log file is used only to recover the database from a system or media failure that prevents database data from being written to a database's datafiles.

For example, if an unexpected power outage abruptly terminates database operation, data in memory cannot be written to the datafiles and the data is lost. After power is restored, any lost data is recovered when the database is opened. By applying the information in the most recent redo log files to the database's datafiles, Oracle restores the database to the time at which the power failure occurred.

The process of applying the redo logs to datafiles and control files in order to recover them is called *rolling forward*.

Datafiles

Every Oracle database has one or more physical *datafiles*. A database's datafiles contain all the database data. The data of logical database structures such as tables and indexes is physically stored in the datafiles allocated for a database.

The following are characteristics of datafiles:

- A datafile can be associated with only one database.

- Database files can have certain characteristics set to allow them to automatically extend when the database runs out of space.
- One or more datafiles form a logical unit of database storage called a tablespace.

The Use of Datafiles

The data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. For example, assume that a user wants to access some data in a table of a database. If the requested information is not already in the memory cache for the database, it is read from the appropriate datafiles and stored in memory.

Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by the DBW0 background process of Oracle.

Rollback Segments

Every database contains one or more rollback segments, which are portions of the database that record the actions of transactions in the event that a transaction is rolled back. You use rollback segments to provide read consistency, rollback transactions, and to put a database in a transaction-consistent state as part of recovery.

The Use of Rollback Segments

Rollback segments are used for a number of functions in the operation of an Oracle database. In general, the rollback segments of a database store the old values of data changed by ongoing transactions (that is, uncommitted transactions). Among other things, the information in a rollback segment is used during database recovery to “undo” any “uncommitted” changes applied from the redo log to the datafiles. Therefore, if database recovery is necessary, the data is in a consistent state after the rollback segments are used to remove all uncommitted data from the datafiles.

Archived Logs

Archived log files are redo logs that have been filled with redo, made inactive and copied or archived to a backup location. You can archive online redo files before reusing them; this creates the archived log. The presence or absence of an archived redo log is determined by the mode that the database is using:

ARCHIVELOG	The filled online redo log files are archived before they are reused in the cycle.
NOARCHIVELOG	The filled online redo log files are not archived.

When in ARCHIVELOG mode, the database can be completely recovered from both instance and disk failure. The database can also be backed up while it is open and available for use. However, additional administrative operations are required to maintain the archived redo log.

Typically, your only recovery option for databases operating in NOARCHIVELOG mode is to restore the whole database. Your only backup option is to back up the database while it is completely closed. Because no archived redo log is created, no extra work is required by the database administrator.

Note: The *only* time you can recover a database while operating in NOARCHIVELOG is when you have not already overwritten the online log files that were current at the time of the most recent backup.

2

What Are You Backing Up?

This chapter describes the structures comprising a database, as well as key backup and recovery concepts, and includes the following topics:

- The Online Redo Log
- The Archived Redo Log
- Database Archiving Modes
- Control Files
- Types of Backups

Recovery processes vary depending on the type of failure that occurred, the structures affected, and the type of backups available for performing recovery. If no files are lost or damaged, recovery may amount to no more than restarting an instance.

Several structures of an Oracle database safeguard data against possible failures. This chapter briefly introduce each of these structures and their use in backup and recovery.

The Online Redo Log

Every instance of an Oracle database has an associated online redo log to protect the database in case the database experiences an instance failure. An online redo log consists of two or more pre-allocated files that store all changes made to the database as they occur.

> **Note:** Oracle does not recommend backing up the online redo log. See "Online Redo Log Backups Not Recommended" on page 2-25 for more information.

Online Redo Log File Contents

Online redo log files are filled with *redo entries*. Redo entries record data that can be used to reconstruct all changes made to the database, including the rollback segments. Therefore, the online redo log also protects rollback data.

> **Note:** Redo entries store low-level representations of database changes that cannot be mapped to user actions. Therefore, the contents of an online redo log file should never be edited or altered, and cannot be used for any application purposes such as auditing.

Redo entries are buffered in a "circular" fashion in the redo log buffer of the SGA and are written to one of the online redo log files by the Oracle background process Log Writer (LGWR). Whenever a transaction is committed, LGWR writes the transaction's redo entries from the redo log buffer of the system global area (SGA) to an online redo log file, and a *system change number* (SCN) is assigned to identify the redo entries for each committed transaction.

However, redo entries can be written to an online redo log file before the corresponding transaction is committed. If the redo log buffer fills, or another transaction commits, LGWR flushes all of the redo log entries in the redo log buffer to an online redo log file, even though some redo entries may not be committed.

How Online Redo Log Files Are Written

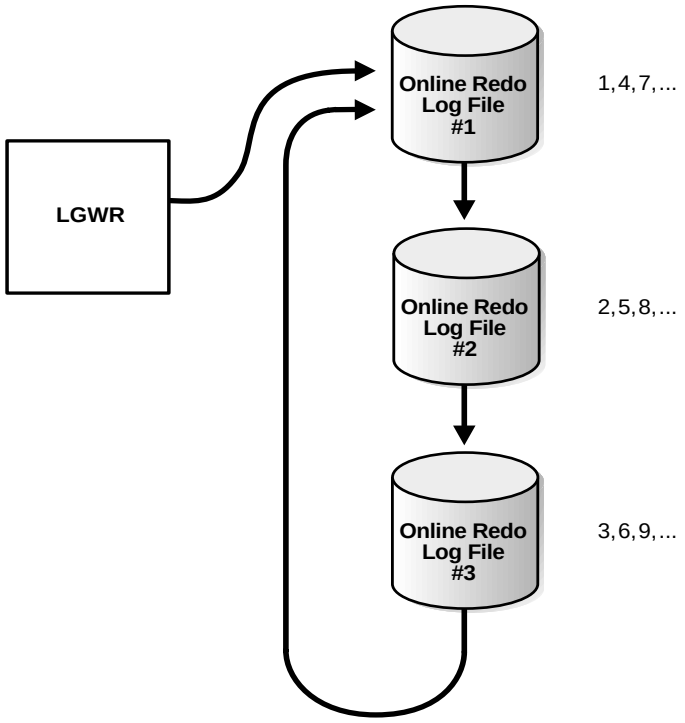
The online redo log of a database consists of two or more online redo log files. Oracle requires two files to guarantee that one is always available for writing while the other is being archived, if desired.

LGWR writes to online redo log files in a circular fashion; when the current online redo log file is filled, LGWR begins writing to the next available online redo log file. When the last available online redo log file is filled, LGWR returns to the first online redo log file and writes to it, starting the cycle again. Figure 2–1 illustrates the circular writing of the online redo log file. The numbers next to each line indicate the sequence in which LGWR writes to each online redo log file.

Filled online redo log files are "available" to LGWR for reuse depending on whether archiving is enabled:

- If archiving is disabled, a filled online redo log file is available once the checkpoint involving the online redo log file has completed.
- If archiving is enabled, a filled online redo log file is available to LGWR once the checkpoint involving the online redo log file has completed *and* once the file has been archived.

Figure 2–1 Circular Use of Online Redo Log Files by LGWR



Active (Current) and Inactive Online Redo Log Files

At any given time, Oracle uses only one of the online redo log files to store redo entries written from the redo log buffer. The online redo log file actively being written by LGWR is called the *current* online redo log file.

Online redo log files that are required for instance recovery are called *active* online redo log files. Online redo log files that are not required for instance recovery are called *inactive*.

If archiving is enabled, an active online log file cannot be reused or overwritten until its contents are archived. If archiving is disabled, when the last online redo log file fills, writing continues by overwriting the first available active file.

Log Switches and Log Sequence Numbers

The point at which Oracle ends writing to one online redo log file and begins writing to another is called a *log switch*. A log switch always occurs when the current online redo log file is completely filled and writing must continue to the next online redo log file. The database administrator can also force log switches.

Oracle assigns each online redo log file a new *log sequence number* every time that a log switch occurs and LGWR begins writing to it. If online redo log files are archived, the archived redo log file retains its log sequence number. The online redo log file that is cycled back for use is given the next available log sequence number.

Each redo log file (including online and archived) is uniquely identified by its log sequence number. During instance or media recovery, Oracle properly applies redo log files in ascending order by using the log sequence number of necessary archived and online redo log files.

Checkpoints

An event called a *checkpoint* occurs when the Oracle background process, DBW0 writes all the modified database buffers in the SGA, including both committed and uncommitted data, to the data files. Checkpoints are implemented for the following reasons:

- Checkpoints ensure that data blocks in memory that change frequently are written to datafiles regularly. Because of the least-recently-used algorithm of DBW0, a data block that changes frequently might never qualify as the least recently used block and thus might never be written to disk if checkpoints did not occur.
- Because all database changes up to the checkpoint have been recorded in the datafiles, redo log entries before the checkpoint no longer need to be applied to the datafiles if instance recovery is required. Therefore, checkpoints are useful because they can expedite instance recovery.

Though some overhead is associated with a checkpoint, Oracle does not halt activity nor are current transactions affected. Because DBW0 continuously writes database buffers to disk, a checkpoint does not necessarily require many data blocks to be written all at once. Rather, the completion of a checkpoint simply guarantees that all data blocks modified since the previous checkpoint are actually written to disk.

Checkpoints occur whether or not filled online redo log files are archived. If archiving is disabled, a checkpoint affecting an online redo log file must complete

before the online redo log file can be reused by LGWR. If archiving is enabled, a checkpoint must complete *and* the filled online redo log file must be archived before it can be reused by LGWR.

Checkpoints can occur for all datafiles of the database (called database checkpoints) or can occur for only specific datafiles. The following list explains when checkpoints occur and what type happens in each situation:

- A database checkpoint automatically starts at every log switch. If a previous database checkpoint is currently in progress, a checkpoint forced by a log switch overrides the current checkpoint.
- An initialization parameter, LOG_CHECKPOINT_INTERVAL, can be set to force a database checkpoint when a predetermined number of redo log blocks have been written to disk relative to the last database checkpoint. You can set another parameter, LOG_CHECKPOINT_TIMEOUT, to force a database checkpoint a specific number of seconds after the previous database checkpoint started. These parameters are useful when extremely large redo log files are used and additional checkpoints are desired between log switches. Database checkpoints signaled to start by these initialization parameters are not performed until the previous checkpoint has completed.
- When the beginning of an online tablespace backup is indicated, a checkpoint is forced *only* on the datafiles that constitute the tablespace being backed up. A checkpoint at this time overrides any previous checkpoint still in progress. Also, since this type of checkpoint only affects the datafiles being backed up, it does not reduce the amount of redo that would be needed for instance recovery.
- If the administrator takes a tablespace offline with normal or temporary priority, a checkpoint is forced *only* on the online datafiles of that tablespace.
- If the database administrator shuts down an instance (NORMAL or IMMEDIATE shutdown transaction), Oracle forces a database checkpoint to complete before the instance is shut down. A database checkpoint forced by instance shutdown overrides any previously running checkpoint.
- The database administrator can force a database checkpoint to happen on demand. A checkpoint forced on demand overrides any previously running checkpoint.

Incremental Checkpointing

Incremental checkpointing improves the performance of crash and instance recovery (but not media recovery). An incremental checkpoint records the position in the redo thread (log) from which crash / instance recovery needs to begin. This log position is determined by the oldest dirty buffer in the buffer cache. The incremental checkpoint information is maintained periodically with minimal or no overhead during normal processing.

Recovery performance is roughly proportional to the number of buffers that had not been written to the database prior to the crash. You can influence the performance of crash or instance recovery by setting the parameter DB_BLOCK_MAX_DIRTY_TARGET, which specifies an upper bound on the number of dirty buffers that can be present in the buffer cache of an instance at any moment in time. Thus, it is possible to influence recovery time for situations where the buffer cache is very large and / or where there are stringent limitations on the duration of crash / instance recovery. Smaller values of this parameter impose higher overhead during normal processing since more buffers have to be written. On the other hand, the smaller the value of this parameter, the better the recovery performance, since fewer blocks need to be recovered.

Incremental checkpoint information is maintained automatically by the Oracle8 server without affecting other checkpoints (such as log switch checkpoints and user-specified checkpoints). In other words, incremental checkpointing occurs independently of other checkpoints occurring in the instance.

Incremental checkpointing is beneficial for recovery in a single instance as well as a multi-instance environment.

Note: Checkpoints also occur at other times if the Oracle Parallel Server is used; see *Oracle8 Parallel Server Concepts and Administration*for more information.

The Mechanics of a Checkpoint

When a checkpoint occurs, the checkpoint background process (CKPT) remembers the location of the next entry to be written in an online redo log file and signals the database writer background process (DBW0) to write the modified database buffers in the SGA to the datafiles on disk. CKPT then updates the headers of all control files and datafiles to reflect the latest checkpoint.

When a checkpoint is not happening, DBW0 only writes the least-recently-used database buffers to disk to free buffers as needed for new data. However, as a checkpoint proceeds, DBW0 writes data to the data files on behalf of both the checkpoint and ongoing database operations. DBW0 writes a number of modified data buffers on behalf of the checkpoint, then writes the least recently used buffers, as needed, and then writes more dirty buffers for the checkpoint, and so on, until the checkpoint completes.

Depending on what signals a checkpoint to happen, the checkpoint can be either "normal" or "fast." With a normal checkpoint, DBW0 writes a small number of data buffers each time it performs a write on behalf of a checkpoint. With a fast checkpoint, DBW0 writes a large number of data buffers each time it performs a write on behalf of a checkpoint.

Therefore, by comparison, a normal checkpoint requires more I/Os to complete than a fast checkpoint. Because a fast checkpoint requires fewer I/Os, the checkpoint completes very quickly. However, a fast checkpoint can also detract from overall database performance if DBW0 has a lot of other database work to complete. Events that trigger normal checkpoints include log switches and checkpoint intervals set by initialization parameters; events that trigger fast checkpoints include online tablespace backups, instance shutdowns, and database administrator-forced checkpoints.

Until a checkpoint completes, all online redo log files written since the last checkpoint are needed in case a database failure interrupts the checkpoint and instance recovery is necessary. Additionally, if LGWR cannot access an online redo log file for writing because a checkpoint has not completed, database operation suspends temporarily until the checkpoint completes and an online redo log file becomes available. In this case, the normal checkpoint becomes a fast checkpoint, so it completes as soon as possible.

For example, if only two online redo log files are used, and LGWR requires another log switch, the first online redo log file is unavailable to LGWR until the checkpoint for the previous log switch completes.

> **Note:** The information that is recorded in the datafiles and control files as part of a checkpoint varies if the Oracle Parallel Server configuration is used; see *Oracle8 Parallel Server Concepts and Administration*.

You can set the initialization parameter LOG_CHECKPOINTS_TO_ALERT to determine if checkpoints are occurring at the desired frequency. The default value of NO for this parameter does not log checkpoints. When you set the parameter to YES, information about each checkpoint is recorded in the ALERT file.

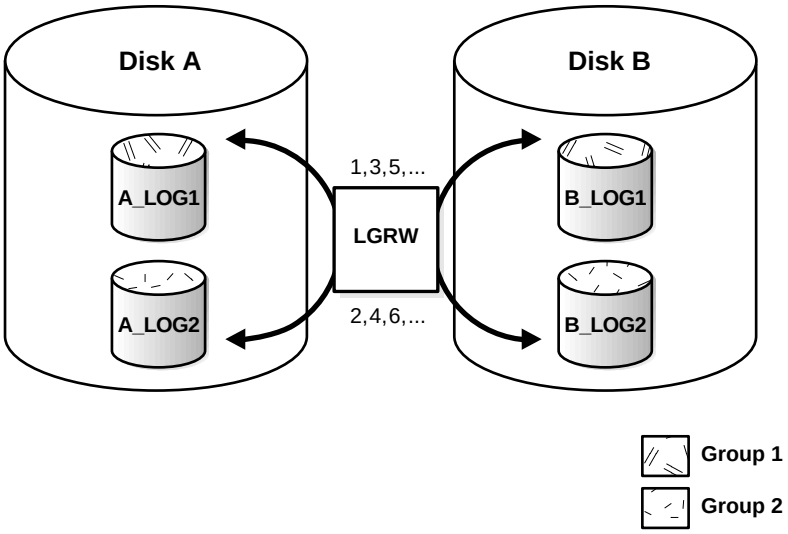
Multiplexed Online Redo Log Files

Oracle provides the capability to *multiplex* an instance's online redo log files to safeguard against damage to its online redo log files. With multiplexed online redo log files, LGWR concurrently writes the same redo log information to multiple identical online redo log files, thereby eliminating a single point of online redo log failure.

> **Note:** Oracle recommends that you multiplex your redo log files; the loss of the log file information can be catastrophic if a recovery operation is required.

Figure 2–2 illustrates duplexed (two sets of) online redo log files.

Figure 2–2 Multiplexed Online Redo Log Files



The corresponding online redo log files are called *groups*. Each online redo log file in a group is called a *member*. Notice that all members of a group are concurrently active (concurrently written to by LGWR), as indicated by the identical log sequence numbers assigned by LGWR. If a multiplexed online redo log is used, each member in a group must be the exact same size.

The Mechanics of a Multiplexed Online Redo Log

LGWR always addresses all members of a group, whether the group contains one or many members. For example, after a log switch, LGWR concurrently writes to all members of the next group, and so on. LGWR never writes concurrently to one member of a given group and one member of another group.

LGWR reacts differently when certain online redo log members are unavailable, depending on the reason for the file(s) being unavailable:

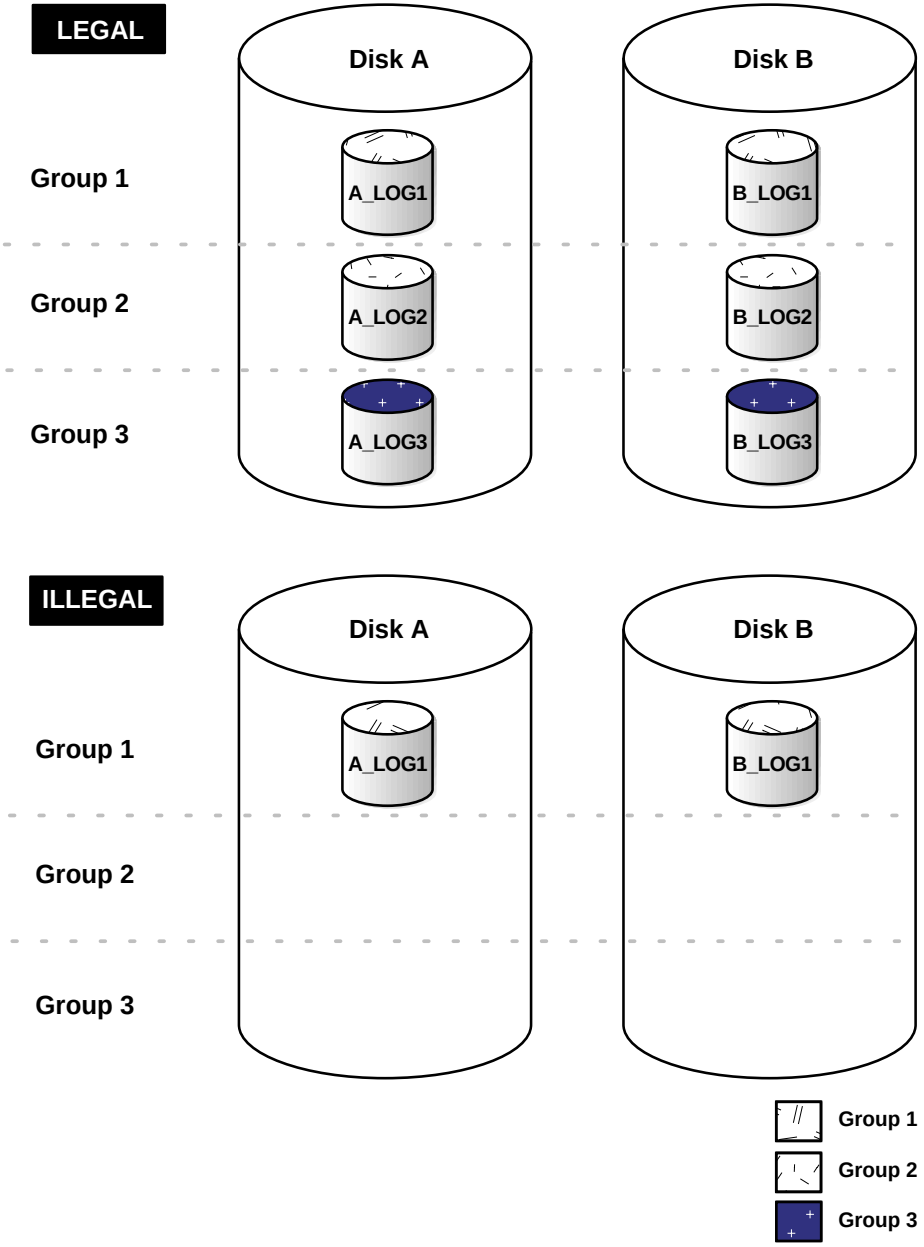
- If LGWR can successfully write to at least one member in a group (either at a log switch or as writing to the group is proceeding), writing to the accessible members of the group proceeds as normal; LGWR simply writes to the available members of a group and ignores the unavailable members.
- If LGWR cannot access the next group at a log switch because the group needs to be archived, database operation is temporarily halted until the group becomes available (in other words, until the group is archived).

- If all members of the next group are inaccessible to LGWR at a log switch because of one or more disk failures, an error is returned and the database instance is immediately shut down. In this case, the database may need media recovery from the loss of an online redo log file. However, if the database checkpoint has moved beyond the lost log (this is not the current log in this example), media recovery is not necessary. Simply drop the inaccessible log group. If the log was not archived, you can disable archiving before the log can be dropped by using the ALTER DATABASE CLEAR UNARCHIVED LOG command.
- If all members of a group suddenly become inaccessible to LGWR as they are being written, an error is returned and the database instance is immediately shut down. In this case, the database might need media recovery from the loss of an online redo log file. If the media containing the log is not actually lost — for example, if the drive for the log was inadvertently turned off — media recovery might not be needed. In this example, all that is necessary is to turn the drive back on and do instance recovery.

Whenever LGWR cannot write to a member of a group, Oracle marks that member as stale and writes an error message to the LGWR trace file and to the database's ALERT file to indicate the problem with the inaccessible file(s).

To safeguard against a single point of online redo log failure, a multiplexed online redo log should be completely symmetrical: all groups of the online redo log should have the same number of members. However, Oracle does not require that a multiplexed online redo log be symmetrical. For example, one group can have only one member, while other groups can have two members. Oracle allows this behavior to provide for situations that temporarily affect some online redo log members but leave others unaffected (for example, a disk failure). The only requirement for an instance's online redo log, multiplexed or non-multiplexed, is that it be comprised of at least two groups. Figure 2–3 shows a legal and illegal multiplexed online redo log configuration. The second configuration is illegal because it has only one group.

Figure 2–3 Legal and Illegal Multiplexed Online Redo Log Configuration



Threads of Online Redo Log and the Oracle Parallel Server

Each database instance has its own online redo log groups. These online redo log groups, multiplexed or not, are called an instance's "thread" of online redo. In typical configurations, only one database instance accesses an Oracle database, thus only one thread is present. However, when running the Oracle Parallel Server, two or more instances concurrently access a single database; each instance in this type of configuration has its own thread.

This manual describes how to configure and manage the online redo log when the Oracle Parallel Server is not used. The thread number can be assumed to be 1 in all discussions and examples of commands. For complete information about configuring the online redo log with the Oracle Parallel Server, see *Oracle8 Parallel Server Concepts and Administration*.

The Archived Redo Log

Oracle allows the optional archiving of filled groups of online redo log files, which creates archived (offline) redo logs. The archiving of filled groups has two key advantages relating to database backup and recovery options:

- A database backup, together with online and archived redo log files, guarantees that all committed transactions can be recovered in the event of an operating system or disk failure.
- A backup taken while the database is open and in normal system use can be used if an archived log is kept permanently.

The choice of whether or not to enable the archiving of filled groups of online redo log files depends on the availability and reliability requirements of the application running on the database. If you cannot afford to lose any data in your database in the event of a disk failure, you must use ARCHIVELOG mode. However, the archiving of filled online redo log files can require the database administrator to perform extra administrative operations.

The Mechanics of Archiving

Depending on how you configure archiving, the mechanics of archiving redo log groups are performed by either the optional Oracle background process ARCH (when automatic archiving is used) or the user process that issues a SQL statement to archive a group manually.

> **Note:** For simplicity, the remainder of this section is based on the assumption that automatic archiving is enabled and the ARCH background process is responsible for archiving filled online redo log groups.

ARCH can archive a group after the group becomes inactive and the log switch to the next group has completed. The ARCH process can access any members of the group, as needed, to complete the archiving of the group. If ARCH attempts to open or read a member of a group and it is not accessible (for example, due to a disk failure), ARCH automatically tries to use another member of the group, and so on, until it finds a member of the group that is available for archiving. If ARCH is archiving a member of a group, and the information in the member is detected as invalid or a disk failure occurs as archiving proceeds, ARCH automatically switches to another member of the group to continue archiving the group where it was interrupted.

A group of online redo log files does not become available to LGWR for reuse until ARCH has archived the group. This restriction is important because it guarantees that LGWR cannot accidentally write over a group that has not been archived, which would prevent the use of all subsequent archived redo log files during a database recovery.

> **Note:** A missing archive log will render all subsequent archive logs useless.

When archiving is used, an archiving destination is specified by setting the parameter LOG_ARCHIVE_DEST. This destination is usually a storage device separate from the disk drives that hold the datafiles, online redo log files, and control files of the database. Typically, the archiving destination is another disk drive of the database server. This way, archiving does not contend with other files required by the instance and completes quickly so the group can become available to LGWR. Ideally, archived redo log files (and corresponding database backups)

should be moved permanently to inexpensive offline storage media, such as tape, that can be stored in a safe place, separate from the database server.

At log switch time, when no more information will be written to a redo log, a record is created in the database's control file. Each record contains the thread number, log sequence number, low SCN for the group, and next SCN after the archived log file; this information is used during database recovery in Parallel Server configurations to automate the application of redo log files.

A record is also created whenever a log is successfully archived. This record contains the name of the archived log as well as the low and high SCN, and log sequence number.

See Also: See *Oracle8 Parallel Server Concepts and Administration* for additional information.

Archived Redo Log File Contents

An archived redo log file is a simple copy of the identical filled members that constitute an online redo log group. Therefore, an archived redo log file includes the redo entries present in the identical members of a group at the time the group was archived. The archived redo log file also preserves the group's log sequence number.

If archiving is enabled, LGWR is not allowed to reuse an online redo log group until it has been archived. Therefore, it is guaranteed that the archived redo log contains a copy of every group (uniquely identified by log sequence numbers) created since archiving was enabled.

Duplexing the Archived Redo Log

It is possible to write out two copies of an online redo log group to archive. Many sites may choose to do this in order to further protect the archived redo log against media failure.

The parameter LOG_ARCHIVE_DUPLEX_DEST determines the second location. Any time a redo log file is archived, it will be archived to both LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST.

If set, the parameter LOG_ARCHIVE_MIN_SUCCEED_DEST determines the number of archive log destinations that a redo log group must be successfully archived to. For additional information see the *Oracle8 Reference*.

Database Archiving Modes

A database can operate in two distinct modes: NOARCHIVELOG mode (media recovery disabled) or ARCHIVELOG mode (media recovery enabled).

NOARCHIVELOG Mode (Media Recovery Disabled)

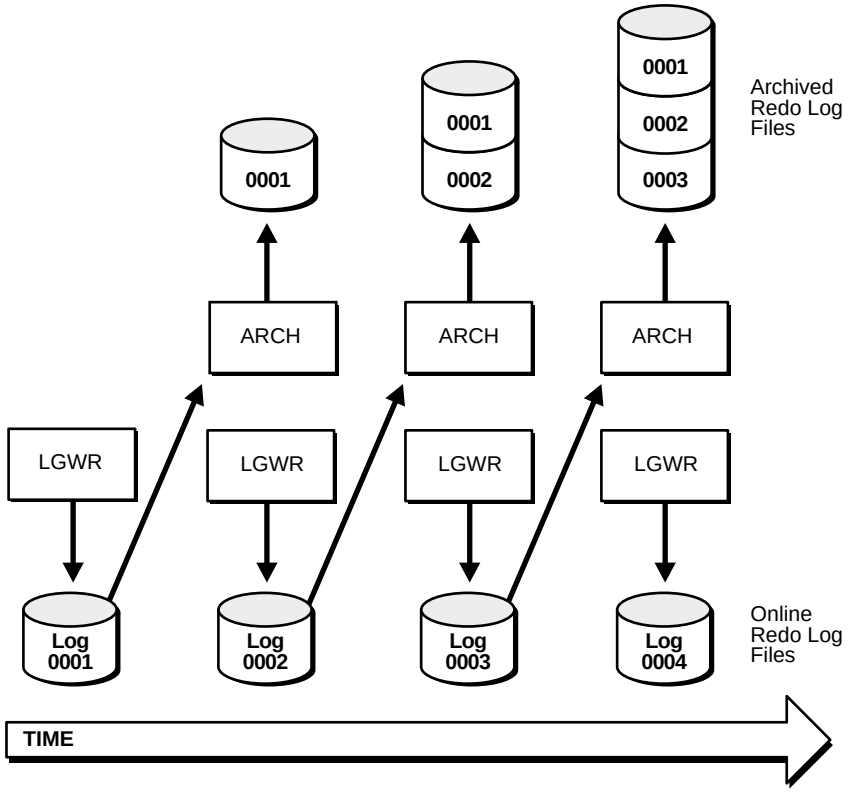
If a database is used in NOARCHIVELOG mode, the archiving of the online redo log is disabled. Information in the database's control file indicates that filled groups are not required to be archived. Therefore, once a filled group becomes inactive and the checkpoint at the log switch completes, the group is available for reuse by LGWR.

NOARCHIVELOG mode protects a database only from instance failure, not from disk (media) failure. Only the most recent changes made to the database, stored in the groups of the online redo log, are available for instance recovery.

ARCHIVELOG Mode (Media Recovery Enabled)

If an Oracle database is operated in ARCHIVELOG mode, the archiving of the online redo log is enabled. Information in a database control file indicates that a group of filled online redo log files cannot be reused by LGWR until the group is archived. A filled group is immediately available to the process performing the archiving once a log switch occurs (when a group becomes inactive); the process performing the archiving does not have to wait for the checkpoint of a log switch to complete before it can access the inactive group for archiving. Figure 2–4 illustrates how the database's online redo log files are used in ARCHIVELOG mode and how the archived redo log is generated by the process archiving the filled groups (for example, ARCH in this illustration).

Figure 2–4 Online Redo Log File Use in ARCHIVELOG Mode



ARCHIVELOG mode permits complete recovery from disk failure as well as instance failure, because all changes made to the database are permanently saved in an archived redo log.

Automatic Archiving and the ARCH (Archiver) Background Process

An instance can be configured to have an additional background process, Archiver (ARCH), automatically archive groups of online redo log files once they become inactive. Therefore, automatic archiving frees the database administrator from having to keep track of, and archive, filled groups manually. For this convenience alone, automatic archiving is the choice of most database systems that have an archived redo log.

If you request automatic archiving at instance startup by setting the LOG_ARCHIVE_START initialization parameter, Oracle starts ARCH during instance startup. Otherwise, ARCH is not started during instance startup.

However, the database administrator can interactively start or stop automatic archiving at any time. If automatic archiving was not specified to start at instance startup, and the administrator subsequently starts automatic archiving, the ARCH background process is created. ARCH then remains for the duration of the instance, even if automatic archiving is temporarily turned off and turned on again.

ARCH always archives groups in order, beginning with the lowest sequence number. ARCH automatically archives filled groups as they become inactive. A record of every automatic archival is written in the ARCH trace file by the ARCH process. Each entry shows the time the archive started and stopped. This information is also recorded in the database's control file, and can be viewed by querying the view V$ARCHIVED_LOG.

If ARCH encounters an error when attempting to archive a group (for example, due to an invalid or filled destination), ARCH continues trying to archive the group. An error is also written in the ARCH trace file and the ALERT file. If the problem is not resolved, eventually all online redo log groups become full, yet not archived, and the system halts because no group is available to LGWR. Therefore, if problems are detected, you should either resolve the problem so that ARCH can continue archiving (such as by changing the archive destination) or manually archive groups until the problem is resolved.

Manual Archiving

If a database is operating in ARCHIVELOG mode, the database administrator can manually archive the filled groups of inactive online redo log files, as necessary, regardless of whether automatic archiving is enabled or disabled. If automatic archiving is disabled, the database administrator is responsible for archiving all filled groups.

For most systems, automatic archiving is chosen because the administrator does not have to watch for a group to become inactive and available for archiving. Furthermore, if automatic archiving is disabled and manual archiving is not performed fast enough, database operation can be suspended temporarily whenever LGWR is forced to wait for an inactive group to become available for reuse.

The manual archiving option is provided so that the database administrator can:

- archive a group when automatic archiving has been stopped because of a problem (for example, the offline storage device specified as archived redo log destination has experienced a failure or become full)
- archive a group in a non-standard fashion (for example, one group to one offline storage device, the next group to a different offline storage device, and so on)
- re-archive a group if the original archived version is lost or damaged

When a group is archived manually, the user process issuing the statement to archive a group actually performs the process of archiving the group. Even if the ARCH background process is present for the associated instance, it is the user process that archives the group of online redo log files.

Control Files

The control file of a database is a small binary file necessary for the database to start and operate successfully. A control file is updated continuously by Oracle during database use, so it must be available for writing whenever the database is mounted. If for some reason the control file is not accessible, the database will not function properly.

Each control file is associated with only one Oracle database.

Control File Contents

A control file contains information about the associated database that is required for the database to be accessed by an instance, both at startup and during normal operation. A control file's information can be modified only by Oracle; no database administrator or end-user can edit a database's control file.

Among other things, a control file contains information such as:

- the database name
- the timestamp of database creation
- the names and locations of associated datafiles and online redo log files
- the current log sequence number
- checkpoint information
- backup information (when using Recovery Manager)

The database name and timestamp originate at database creation. The database's name is taken from either the name specified by the initialization parameter DB_NAME or the name used in the CREATE DATABASE statement.

Each time that a datafile or an online redo log file is added to, renamed in, or dropped from the database, the control file is updated to reflect this physical structure change. These changes are recorded so that:

- Oracle can identify the datafiles and online redo log files to open during database startup.
- Oracle can identify files that are required or available in case database recovery is necessary.

Therefore, if you make a change to your database's physical structure, you should immediately make a new backup of your control file.

Control files also record information about checkpoints. When a checkpoint starts, the control file records information to remember the next entry that must be entered into the online redo log. This information is used during database recovery to tell Oracle that all redo entries recorded before this point in the online redo log group are not necessary for database recovery; they were already written to the datafiles.

Multiplexed Control Files

As with online redo log files, Oracle allows multiple, identical control files to be open concurrently and written for the same database. By storing multiple control files for a single database on different disks, you can safeguard against a single point of failure with respect to control files. If a single disk containing a control file crashes, the current instance fails when Oracle attempts to access the damaged control file. However, if the control file is multiplexed, other copies of the current control file are available on different disks. This allows the instance to be restarted easily without the need for database recovery.

The permanent loss of all copies of a database's control file is a serious problem to safeguard against. If *any copy of a control file fails* during operation, the instance is aborted and media recovery is required. If the control file is not multiplexed, then the recovery procedure will be more complex. Therefore, it is strongly recommended that multiplexed control files be used with each database, with each copy stored on a different physical disk.

Types of Backups

This section defines and describes the following types of backups:

- Whole Database Backups
  - Consistent Whole Database Backups
  - Inconsistent Whole Database Backups
- Tablespace Backups
- Datafile Backups
- Controlfile Backups
- Archivelog Backups

Logical Backups, also known as Exports, are described in detail in *Oracle8 Utilities*.

Whole Database Backups

A whole database backup contains the control file, and all database files which belong to that database. Whole database backups are the most common type of backups performed by database administrators. Before performing whole database backups (or any other type of backup), you should first be familiar with your enterprise's backup strategy. Specifically, you should be aware of the implications of backing up when in ARCHIVELOG mode and NOARCHIVELOG mode

During a whole database backup, all datafiles and control files belonging to that database are backed up (you should not need to back up the online redo logs).

Whole database backups do not require the database to be operated in a specific archiving mode; they can be taken whether a database is operating in ARCHIVELOG or NOARCHIVELOG mode.

If the database is in ARCHIVELOG mode, you can choose to back up the database while it is open or closed. A database running in NOARCHIVELOG mode should only be backed up when it is closed by a clean shutdown (for example, a shutdown using the immediate or normal options). *A backup of a database running in NOARCHIVELOG mode and not shutdown cleanly is useless.* In such cases, the backed up files are inconsistent with respect to a point-in-time, and because the database is in NOARCHIVELOG mode, there are no logs available to make the database consistent. Recovery Manager does not allow you to back up a database that has been running in NOARCHIVELOG mode and shutdown abnormally, because the backup is not usable for recovery.

You should always back up the parameter files associated with the database, as well as the instance's password file (if the instance uses password files) whenever they are modified.

> **Note:** In ARCHIVELOG mode, a whole database backup can be constructed using backups of datafiles taken at different times. For example, if a database is composed of seven tablespaces, and each night a different tablespace is backed up (the control file is backed up every night), after seven nights all tablespaces in the database, and the control file will have been backed up—this can be considered a whole database backup.

There are two types of whole database backups: consistent and inconsistent.

Consistent Whole Database Backups

A *consistent* whole database backup is a backup where all files (datafiles and control files) are consistent to the same point in time (for example, checkpointed at the same SCN). The only tablespaces in a consistent backup that are allowed to have older timestamps (SCNs) are read-only and offline normal tablespaces, which are still consistent with the other datafiles in the backup.

You can open the set of files resulting from a consistent whole database backup without applying redo logs because the data is already consistent; no action is required to make the data in the restored datafiles correct.

A consistent whole database backup is the only valid backup option for databases running in NOARCHIVELOG mode. The only way to take a consistent whole database backup is to shut down the database cleanly and take the backup while the database is closed.

To make a consistent database backup current (or to take it to a later point in time) you will either need to apply redo to it, or, if you are using Recovery Manager to perform your backups, you may have the option of applying a combination of incremental backups and redo. The redo is located in the archived logs, and the online logs (if you are recovering to the current time).

If a database is not shut down cleanly (for example, an instance failure occurred, or a SHUTDOWN ABORT statement was issued), the database's datafiles will most likely be inconsistent.

> **WARNING:** A backup control file created during a consistent whole database backup should only be used if you are restoring the whole database backup and do not intend to perform any recovery. If you intend to perform recovery and have a current control file, you should not restore an older control file.

Inconsistent Whole Database Backups

If you are in a situation where your database must be up and running 24 hours a day, 7 days a week, you will need to perform *inconsistent* whole database backups. A backup of an open database is inconsistent because portions of the database (hence, datafiles in the database) are being modified and written to disk while the backup is progressing. The database must be in ARCHIVELOG mode to be able to perform open backups.

Deciding whether or not to perform an open backup depends only upon the availability requirements of your data—open backups are the only choice if the data being backed up must always be available.

Inconsistent backups are also created if a database is backed up after a system crash failure or shutdown abort. This type of backup is valid if the database is running in ARCHIVELOG mode, because all logs are available to make the backup consistent.

> **Note:** Oracle recommends that you do not take inconsistent, closed database backups.

After open (also called "hot") or inconsistent closed backups, you should always archive the current online log file. Archive the current online log by issuing the SQL command ALTER SYSTEM ARCHIVE ALL. After archiving the current log file, back up all archivelogs produced since the backup began; this ensures that you can use the backup. If you do not have all archived logs produced during the backup, the backup cannot be recovered (because you do not have all the logs necessary to make the it consistent).

Unless you are taking a consistent whole database backup, you should back up your control file using the ALTER DATABASE command with the BACKUP CONTROLFILE option. A database must be consistent before it can be opened.

Inconsistent whole database backups are made consistent by applying incremental backups and redo logs (archived and online).

> **Note:** Inconsistent whole database backups of databases running in NOARCHIVELOG mode are usable only if the logs relating to that backup are available. For databases running in NOARCHIVELOG mode, you should aim to have a backup that is usable without performing any recovery. This aim is defeated if you need to apply redo from logs to recover a backup. If you need redo to make a database consistent, the database should really operate in ARCHIVELOG mode, because this is the way online logs are backed up.

Tablespace Backups

A tablespace backup is a backup of a subset of the database.

Tablespace backups are only valid if the database is operating in ARCHIVELOG mode. The only time a tablespace backup is valid for a database running in NOARCHIVELOG mode is when that tablespace (and all datafiles in that tablespace), is read-only or offline-normal.

Datafile Backups

A datafile backup is a backup of a single datafile. Database administrators usually take tablespace backups rather than datafile backups, because the tablespace is a logical unit of a database to back up.

Datafile backups are only valid if the database is operating in ARCHIVELOG mode. The only time a datafile backup is valid for a database running in NOARCHIVELOG mode is if that datafile is the only file in a tablespace. For example, the backup is a tablespace backup, but the tablespace only contains one file and is read-only or offline-normal.

Control File Backups

A control file backup is a backup of a database's control file. If the database is open you can issue the following statement:

```
ALTER DATABASE BACKUP CONTROLFILE to 'location';
```

You can also use Recovery Manager to back up the control file.

If you are going to take an O/S backup of the control file, you must shut down the database first.

Archivelog Backups

If a database is operating in ARCHIVELOG mode, online logs are archived to the LOG_ARCHIVE_DEST, from where they are typically backed up to tertiary storage (such as magnetic tape).

Some sites require very fast recovery times, so they may make a copies of their archivelogs to another disk filesystem, as well as to tertiary storage. You can set the parameter LOG_ARCHIVE_DUPLEX_DEST so that it will archive a second copy of each online log to this location. The on-disk archivelog copies are typically kept for a short time (for example, 48 hours) before being deleted.

See Also: "Duplexing the Archived Redo Log" on page 2-15.

The frequency of archivelog backups depends on:

- the number and size of archivelogs produced
- the size of the filesystem that the logs are archived to

Online Redo Log Backups Not Recommended

The best way to back up the contents of the current online log is to archive it, then back up the archived log.

Oracle recommends you do not copy a current online log, because when you do so, and then restore that copy, the copy will appear as the end of the redo thread. However, additional redo may have been generated in the thread. If you ever attempt to execute recovery supplying the redo log copy, recovery will erroneously detect the end of the redo thread and prematurely terminate, possibly corrupting the database.

Backup Formats

Each of the backup types described in the preceding sections can be made in the following formats:

- Backup Sets
  - Datafiles
  - Archived Redo Log Files
- Datafile Copies
- Operating System Backups

Backup Sets

Backup sets are created using the Recovery Manager **backup** command, and can contain either archive logs or datafiles, but not both.

An Oracle server process reads the datafiles being backed up and creates the backup set. This is why backup sets of open database files (made by Recovery Manager) do *not* need to be preceded by the ALTER TABLESPACE BEGIN BACKUP statement.

For additional information see “Fractured Block Detection During Open Database Backups in Recovery Manager” on page 7-26.

Backup sets are written out in an Oracle proprietary format. Files in a backup set cannot be used by an Oracle instance until they are restored to an instance-usable (usually datafile) format by Recovery Manager. For example, a tablespace backup in a backup set is a compressed version of each file in the tablespace and the Recovery Manager command restores the datafiles from the backup set to an instance-usable format.

You can back up any logical unit of an Oracle database in a backup set:

- If you direct Recovery Manager to perform a whole database backup, it will automatically back up every file in that database, as well as the control file.
- If you direct Recovery Manager to perform a tablespace backup, it will back up each datafile in that tablespace.
- If you direct Recovery Manager to perform a datafile backup, it will back up the datafiles you specify.
- If you direct Recovery Manager to perform an ARCHIVELOG backup, it will back up the archivelogs you specify (this could be ‘all’ archivelogs generated).

You can also direct Recovery Manager to include a control file backup in any datafile backup set.

> **Note:** Archivelogs and datafiles cannot be combined in the same backup set.

Datafile Copies

A datafile copy is created using the Recovery Manager **copy** command.

An Oracle server process reads the datafile and writes the copy out to disk, not an O/S routine. This is why datafile copies of open database files (made by Recovery Manager) do *not* need to be preceded by the ALTER TABLESPACE BEGIN BACKUP statement.

See Also: "Fractured Block Detection During Open Database Backups in Recovery Manager" on page 7-26.

Datafile copies can be used immediately by an Oracle instance—they are already in an instance-usable (usual datafile) format.

> **Note:** Datafile copies can only be made to disk.

Operating System Backups

Operating system (O/S) backups are created using an operating system command. O/S backups can be written to disk or tape in any format that your O/S utilities support.

Recovery Manager can catalog and use O/S backups that are image backups on disk.

Logical Backups

Logical backups store information about the schema objects created for a database. You can use the Export utility to write data from an Oracle database to operating system files that have an Oracle database format.

Because the Oracle Export utility can selectively export specific objects or portions of an object (for example, partitioned tables), you might consider exporting portions or all of a database for supplemental protection and flexibility in a database's backup strategy. Database exports are not a substitute for physical backups and cannot provide the same complete recovery advantages that the built-in functionality of Oracle offers.

See Also: For more information about the Export Utility, see the *Oracle Server Utilities Guide*.

3

When to Perform Backups

This chapter offers guidelines and strategies to follow when planning backups, and includes the following topics:

- Guidelines for Database Backups
- Creating a Backup Strategy
- Backing Up Online Redo Logs

Guidelines for Database Backups

This section describes guidelines that can help you decide when to perform database backups, what parts of a database need backing up, and includes the following topics:

- Perform Backups Frequently and Regularly
- Backup Appropriate Portions of the Database When Making Structural Changes
- Back Up Often-used Tablespaces Frequently
- Keep Older Backups
- Database Backups After Resetlogs
- Export Database Data for Added Protection and Flexibility
- Consider Distributed Database Backups
- Back Up after Performing Unrecoverable / Unlogged Operations
- Test Backup and Recovery Strategies

Before you create an Oracle database, you should decide how you plan to protect the database against potential disk failures, and whether or not to enable point-in-time recovery. If such planning is not considered before database creation, database recovery may not be possible if a disk failure damages the datafiles, online redo log files, or control files of a database.

Perform Backups Frequently and Regularly

Frequent and regular whole database or tablespace backups are essential for any recovery scheme. The frequency of backups should be based on the rate or frequency of changes to database data (such as insertions, updates, and deletions of rows in existing tables, and addition of new tables). If a database's data is changed at a high rate, database backup frequency should be proportionally high. Alternatively, if a database is mainly read-only, and updates are issued only infrequently, the database can be backed up less frequently.

Backup Appropriate Portions of the Database When Making Structural Changes

If you make any of the following structural changes, perform a backup of the appropriate portion of your database immediately before and after completing the alteration:

- create or drop a tablespace
- add or rename (relocate) a datafile in an existing tablespace
- add, rename (relocate), or drop an online redo log group or member

Backing up the appropriate portion of the database depends on the archiving mode of the database, as described below:

- If a database is operated in ARCHIVELOG mode, only a control file backup (using the ALTER DATABASE command with the BACKUP CONTROLFILE option) is required before and after a structural alteration. However, you can back up other parts of the database.
- If the database is operated in NOARCHIVELOG mode, a consistent whole database backup should be taken immediately before and after the modification, including all datafiles and control files.

Back Up Often-used Tablespaces Frequently

If a database is operated in ARCHIVELOG mode, it is acceptable to back up the datafiles of an individual tablespace or even a single datafile. This option is useful if a portion of a database is used more extensively than others, such as the SYSTEM tablespace and tablespaces that contain rollback segments. By taking more frequent backups of the extensively used datafiles of a database, you gather more recent copies of the datafiles. As a result, if a disk failure damages the extensively used datafiles, the more recent backup can restore the damaged files. Only a small number of changes to data need to be applied to roll the restored file forward to the time of the failure, or desired point-in-time recovery, thereby reducing database recovery time.

Back Up after Performing Unrecoverable/Unlogged Operations

If users are creating tables or indexes using the UNRECOVERABLE option, consider taking backups after the objects are created. When tables and indexes are created as UNRECOVERABLE, no redo is logged, and these objects cannot be recovered from existing backups.

> **Note:** If using Recovery Manager, you can take an incremental backup.

See Also: For information about the UNRECOVERABLE option, see the CREATE TABLE...AS SELECT and CREATE INDEX commands in the *Oracle8 SQL Reference*.

Keep Older Backups

How long you should keep an older database backup depends on the choices you want for database recovery. If you want to recover to a past time, you need a database backup which completed before that time. For a database operating in NOARCHIVELOG mode, this means a consistent whole database backup. For a database operating in ARCHIVELOG mode, this means a whole database backup which need not be consistent, which completed before that time (the control file should reflect the database's structure at the point-in-time of recovery), and all archived logs necessary to recover the datafiles to the required point-in-time.

For added protection, consider keeping two or more backups (and all archive logs that go with these backups) previous to the current backup. Thus, if your most recent backups are not usable (for example, the tape drive used for backups writes bad backups), you will not lose all of your data.

> **WARNING:** After opening the database with the RESETLOGS option, existing backups cannot be used for subsequent recovery beyond the time when the logs were reset. You should therefore shutdown the database and make a consistent whole database backup. Doing so will enable recovery of database changes subsequent to using the RESETLOGS option.

Database Backups After Using the RESETLOGS Option

After you have opened a database with the RESETLOGS option, Oracle recommends that you immediately perform a whole database backup. If you do not, and a disaster occurs, you will lose all work performed since opening the database (because it requires a restore of all or part of your database).

> **Note:** There is one exception to this rule. See "Recovery After Using the RESETLOGS Option" on page 3-10 for details.

When a database is opened with the RESETLOGS option, Oracle automatically:

- creates a new "incarnation" of the database
- resets the log sequence number to 1
- reformats the online redo log files if they exist (otherwise they are created)

Oracle performs these actions so that it can identify which archived redo logs apply to which incarnations of the database.

Backup Options While in NOARCHIVELOG mode

If you are operating in NOARCHIVELOG mode, your only option is to make a cold consistent whole database backup.

Backup Options While in ARCHIVELOG mode

If you are operating in ARCHIVELOG mode, you can either perform a consistent (closed) whole database backup, or an inconsistent (open) database backup. The option you choose depends on:

- the amount of time before the database absolutely must be available
- the importance of data entered after opening the database with the RESETLOGS option

If your most important criteria is getting the database up and available, your only option is to perform open database backups.

There is a risk in performing an open database backup— if the backups do not complete before you have another media failure, you will lose all changes made since opening the database with the RESETLOGS option (you cannot use a backup

taken before opening the database with RESETLOGS to recover this incarnation of the database).

> **Note:** There is one and only one exception to this. See "Recovery After Using the RESETLOGS Option" on page 3-10.

If the most important criteria is to restore in case of another failure, then you may elect to take a consistent (closed) database backup.

Time permitting, Oracle recommends that you perform a consistent whole database backup.

Export Database Data for Added Protection and Flexibility

Because the Oracle Export utility can selectively export specific objects, you might consider exporting portions or all of a database for supplemental protection and flexibility in a database's backup strategy. Database exports are not a substitute for whole database backups and cannot provide the same complete recovery advantages that the built-in functionality of Oracle offers.

See Also: For more information on the Export utility, see the *Oracle8 Utilities* guide.

Consider Distributed Database Backups

If a database is a node in a distributed database, consider the following guidelines:

- All databases in the distributed database system should be operated in the same archiving mode.
- If the databases in a distributed database system are operating in ARCHIVELOG mode, backups at each node can be performed autonomously (individually, without time coordination).
- If the databases in a distributed database system are operating in NOARCHIVELOG mode, consistent whole database backups must be performed at the same (global) time, to plan for global distributed database recovery. For example, if a database in New York is backed up at midnight EST, the database in San Francisco should be backed up at 9 PM PST.

Test Backup and Recovery Strategies

Test your backup and recovery strategies in a test environment before and after you move to a production system. By doing so, you can test the thoroughness of your strategies and minimize problems before they occur in a real situation.

Performing test recoveries regularly ensures that your archiving, backup, and recovery procedures work. It also helps you stay familiar with recovery procedures, so that you are less likely to make a mistake in a crisis.

Creating a Backup Strategy

Before you create an Oracle database, decide how you plan to protect the database against potential failures. Answer the following questions before developing your backup strategy:

- **Is it acceptable to lose any data if a disk failure damages some of the files that constitute a database?** If it is not acceptable to lose any data, the database must be operated in ARCHIVELOG mode, ideally with a multiplexed online redo log. If it is acceptable to lose a limited amount of data if there is a disk failure, you can operate the database in NOARCHIVELOG mode and avoid the extra work required to archive filled online redo log files.
- **Will you ever need to recover to arbitrary past points in time?** If you need to recover to a past point in time to correct an erroneous operational or programmatic change to the database, be sure to run in ARCHIVELOG mode and perform control file backups whenever making structural changes. Recovery to a past point in time is facilitated by having a backup control file that reflects the database structure at the desired point-in-time.
- **Does the database need to be available at all times (twenty-four hours per day, seven days per week)?** If so, do not operate the database in NOARCHIVELOG mode because the required whole database backups, taken while the database is shutdown, cannot be made frequently, if at all. Therefore, high-availability databases always operate in ARCHIVELOG mode to take advantage of open datafile backups.

Backup Strategies in NOARCHIVELOG Mode

If a database is operated in NOARCHIVELOG mode, filled groups of online redo log files are not archived. Therefore, the only protection against a disk failure is the most recent whole backup of the database.

Plan to take whole database backups regularly, according to the amount of work that you can afford to lose. For example, if you can afford to lose the amount of work accomplished in one week, make a whole database, closed backup once per week. If you can afford to lose only a day's work, make a whole database, closed backup every day. For large databases with a high amount of activity, it is usually unacceptable to lose work. Therefore, the database should be operated in ARCHIVELOG mode, and the appropriate backup strategies should be used.

Whenever you alter the physical structure of a database operating in NOARCHIVELOG mode, immediately take a consistent whole database backup. A whole database backup fully reflects the new structure of the database.

Backup Strategies in ARCHIVELOG Mode

If a database is operating in ARCHIVELOG mode, filled groups of online redo log files are being archived. Therefore, the archived redo log coupled with the online redo log and datafile backups can protect the database from a disk failure, providing for complete recovery from a disk failure to the instant that the failure occurred (or, to the desired past point-in-time). Following are common backup strategies for a database operating in ARCHIVELOG mode:

- When the database is initially created, perform a whole database, closed backup of the entire database. This initial whole database backup is the foundation of your backups because it provides backups of all datafiles and the control file of the associated database.

  Note: When you perform this initial whole database backup, make sure that the database is in ARCHIVELOG mode first. Otherwise, the backed up control files will contain the NOARCHIVELOG mode setting.

- Subsequent whole database backups are not required, and if a database must remain open at all times, whole database, closed backups are not feasible. Instead, you can take open database or tablespace backups to keep your database backups up-to-date.
- Take open or closed datafile backups to update backed up information for the database (supplementing the initial whole database backup). In particular, the datafiles of extensively used tablespaces should be backed up frequently to reduce database recovery time, should recovery ever be required. If a more recent datafile backup restores a damaged datafile, less redo (or incremental

backups) need to be applied to the restored datafile to roll it forward to the time of the failure.

Whether you should take open or closed datafile backups depends on the availability requirements of the data. Open datafile backups are the only choice if the data being backed up must always be available.

You can also use a datafile copy, taken while the database is open and the tablespace is online, to restore datafiles. However, the data in the restored datafiles is inconsistent. Therefore, the appropriate redo log files (online and archived) must be reapplied to these restored datafiles to make the data consistent and bring it forward to the specified point in time.

- Every time you make a structural change to the database, take a control file backup. If operating in ARCHIVELOG mode and the database is open, use either Recovery Manager or the ALTER DATABASE command with the BACKUP CONTROLFILE option.

  Note: After backing up the control file, apply redo to it up until the point when the datafile was added. Then issue an ALTER DATABASE CREATE DATAFILE statement, and continue with recovery. Do not use operating system utilities to back up the control file in ARCHIVELOG mode, unless you are performing a closed backup.

Backing Up Online Redo Logs

You should never back up online logs for the following reasons:

- The way you protect the online logs against media failure is by multiplexing them (having multiple log members per group, on different disks and different disk controllers).
- If your database is in ARCHIVELOG mode, the logs are being backed up when they are archived.
- If your database is in NOARCHIVELOG mode, the only type of backups that should be performed are closed, consistent, whole database backups. To make the backup consistent, the database must be closed by either a shutdown immediate, shutdown transactional, or shutdown normal. The files in this type of backup are all consistent, and do not need recovery, hence the online logs are not needed.

If a database crashes, and all multiplexed copies of the online redo log are lost, then:

- If your database is in ARCHIVELOG mode, you should restore the whole database and recover the database up until the last archived log. You must then open the database with the RESETLOGS option.

  Having an old backup of online logs in this situation is of no use because the information in the online logs will not be needed for recovery (in fact, the information in the online log files already exists in an archived log).

- If your database is in NOARCHIVELOG mode, your backup method should be to perform consistent whole database backups. The database can then be opened with the RESETLOGS option; no recovery is required because the database is already consistent.

  Note: Recovery Manager will not backup a NOARCHIVELOG mode database unless it has been shutdown cleanly.

Recovery After Using the RESETLOGS Option

In releases prior to Oracle8, DBAs typically backed up online logs when performing cold consistent backups to avoid opening the database with the RESETLOGS option (if they were planning to restore immediately). A classic example of this occurred during disk maintenance, which requires the database to be backed up, deleted, the disks reconfigured, and the database restored. If you avoided RESETLOGS, you would not have to perform a whole database backup immediately after the database was restored. The backup was required since it was impossible to perform recovery using a backup taken before using RESETLOGS (especially if any errors occurred after resetting the logs).

This is no longer the case. There is now only one situation where—if you have a *consistent* backup of the database, taken *immediately before* you open the database with the RESETLOGS option, *and* a control file that is valid *after* you open the database with RESETLOGS—that you can still use this backup to roll forward.

The following scenario illustrates a situation when this is possible.

The DBA wishes to perform hardware striping reconfiguration, which requires the database files to be backed up and deleted, the hardware reconfigured, and the database restored.

On Friday night the DBA performs the following actions:

- Cleanly shut down the database (SHUTDOWN IMMEDIATE)
- Whole database backup (control files and datafiles)

> **Note:** At this point you (the DBA) must not reopen the database.

- O/S maintenance
- Restore database and control files
- STARTUP MOUNT DATABASE
- OPEN DATABASE RESETLOGS

On Saturday morning the following occurs:

- Scheduled batch jobs run, generating a lot of records in the redo log

If a hardware error occurs on Saturday night, and requires restoration of the whole database, it is possible to restore the backup taken immediately before opening the database with the RESETLOGS option, and to roll forward using the logs produced on Saturday.

On Saturday night the following takes place:

- SHUTDOWN ABORT INSTANCE (if it still exists)
- Restore all damaged files from the backup made on Friday night

> **Note:** If you have the current control file, *do not* restore it; otherwise you must restore a control file that was valid after opening the database with RESETLOGS.

- Roll forward

> **WARNING:** It is possible to recover after opening a database with the RESETLOGS option *only* if you have:
>
> - A consistent backup taken *immediately before opening the database*, that is, you must not have opened the database subsequent to the backup and before opening it with the RESETLOGS option.
> - A control file that was valid after opening the database with the RESETLOGS option.

In this scenario, if the DBA had opened the database after the Friday night backup and before opening the database with RESETLOGS, or, did not have a control file from after opening the database, the DBA *would not* be able to use the Friday night backup to roll forward. The DBA must have a backup after opening the database with the RESETLOGS option in order to be able to recover.

It is always good practice to perform a complete backup of your database after opening a database with the RESETLOGS option.

See Also: For additional information see "Database Backups After Using the RESETLOGS Option" on page 3-5.

The Dangers Associated with Backing Up Online Redo Logs

There is no real danger in backing up online logs. The real danger is that you may accidentally restore them while not intending to. There are a number of situations where restoring the online logs would cause very significant problems in the recovery process. Following are two scenarios that illustrate how restoring backed up online logs severely compromises recovery.

Unintentional Restore of Online Redo Logs

When a crisis occurs, it is easy to make a simple mistake. DBAs and system administrators frequently encounter dangers is during a database restore. When restoring the whole database, it is possible to also accidentally restore the online redo logs, thus overwriting the current logs with the older (and useless) backups. This action forces the DBA to perform an incomplete recovery, when the intention was to perform a complete recovery, and ultimately losing the ability to recover valuable transactions.

Erroneously Creating Multiple Parallel Redo Log Time Lines

You can unintentionally create multiple parallel redo log timelines for a single instance database. However, you can avoid this mistake if the online logs cannot be restored (the database must be opened with the RESETLOGS option, which effectively creates the new logs, and also a new database incarnation).

If you faces a problem where the best course of action is to restore the database from a consistent backup (and not perform any recovery) you may think it is safe to restore the online logs, and avoid opening the database with the RESETLOGS option.

If you do this, the database will generate a log sequence number that was already generated by the database during the previous timeline. If you then face another disaster and need to restore from this backup and roll forward, it would be difficult to identify which log sequence number is actually the correct one to apply. In this example, if the logs had been reset, a new incarnation of the database would be created. Any logs created by this new incarnation could only be applied to this incarnation.

Note: Recovery Manager and EBU do not back up online logs for the reasons given in these scenarios.

4

Choosing Recovery Strategies

This chapter offers guidelines and strategies to follow when preparing for database recovery, and includes the following topics:

- Recovery Concepts and Strategies
- Recovery Planning

Recovery Concepts and Strategies

Before recovering a database, familiarize yourself with the fundamental data structures, concepts and strategies of Oracle recovery. This section describes basic recovery issues, and includes the following topics:

- Important Recovery Data Structures
- Recovery Planning
- Recovery Operations

Of course, before learning about the data structures important to recovery operations, you should first understand what is meant by restoring and recovering data.

You *restore* a file by reconstructing or retrieving an original copy of it from a backup.

When you *recover* a file, you are making a restored file current, or current to a specific point in time. This process of recovery is also known as "'rolling forward."

There are two types of recovery commands:

- SQL

  The SQL RECOVER command rolls a restored file forward by applying redo (archived and online redo logs).

- Recovery Manager

  A Recovery Manager **recover** command rolls a restored datafile forward by applying incremental backups (if they exist) and then applying redo (archived and online redo logs).

Important Recovery Data Structures

Table 4–1 describes important data structures involved in recovery processes. Be familiar with these data structures before starting any recovery procedure.

Table 4–1 Recovery Data Structures

Data Structure	Description
Control File	The control file contains records that describe and maintain information about the physical structure of a database. The control file is updated continuously during database use, and must be available for writing whenever the database is open. If the control file is not accessible, the database will not function properly.
System Change Number (SCN)	The system change number is a clock value for the Oracle database that describes a committed version of the database. The SCN functions as a sequence generator for a database, and controls concurrence and redo record ordering. Think of the SCN as a timestamp that helps ensure transaction consistency.
Rollback Segments	Information in a rollback segment is used during database recovery to undo any uncommitted changes applied from the redo log to the datafiles. After the rollback segments are used to remove all uncommitted data from the datafiles, data is in a consistent state.
Redo Records	A redo record is a group of change vectors describing a single, atomic change to the database. Redo records are constructed for all data block changes and saved on disk in the redo log. Redo records allow multiple database blocks to be changed so that either all changes occur or no changes occur, despite arbitrary failures.
Redo Logs	All changes to the Oracle database are recorded in redo logs, which consist of at least two redo log files that are separate from the datafiles. During database recovery from an instance or media failure, Oracle applies the appropriate changes in the database's redo log to the datafiles; this updates database data to the instant that the failure occurred.

Table 4–1 Recovery Data Structures (Cont.)

Backup	A database backup is a copy of the physical files that constitute the Oracle database. When original data is lost, you can use the backup to reconstruct the lost information.
Incremental Backups (Recovery Manager only)	An incremental backup consists of one or more datafiles containing only those blocks that have been modified since a previous backup.
Recovery Catalog (Recovery Manager only)	A repository of information used and maintained by Recovery Manager. Contains information about the following: datafile and archivelog backup sets and pieces datafile copies archived redo logs and copies of them tablespaces and datafiles at the target database user-created stored scripts
Checkpoint	A checkpoint is a data structure in the control file that defines a consistent point of the database across all threads of a redo log. Checkpoints are similar to SCNs, and also describe which threads exist at that SCN. Checkpoints are used by recovery to ensure that Oracle starts reading the log threads for the redo application at the correct point. For Parallel Server, each checkpoint has its own redo information.

Recovery Planning

Before recovering a database, you should create a recovery plan or strategy. This section describes important issues to consider when defining your plan.

Factors Determining Your Recovery Strategy

Your recovery strategy will depend upon the answers to the following questions:

- Is the Database Running in ARCHIVELOG Mode?
- What Files Were Lost or Damaged?
- Has a User Accidentally Destroyed Data?
- Have You Tested Backup and Recovery Strategies?

After answering these questions, you can choose a recovery strategy that is appropriate for your particular situation.The following sections describe the types of recovery to perform in specific situations.

Is the Database Running in ARCHIVELOG Mode?

If your database is *not* running in ARCHIVELOG mode, you should restore the database from a consistent database backup. This is your only option when the database is not in ARCHIVELOG mode. The control file and all datafiles are restored from a consistent backup and the database is opened. All changes made subsequent to the backup are lost.

> **Note:** The *only* time you can recover a database while operating in NOARCHIVELOG is when you have not already overwritten the online log files that were current at the time of the most recent backup.

What Files Were Lost or Damaged?

If one or more datafiles are lost, but the database stays open, you should perform tablespace (or datafile) recovery while the database is open. The tablespaces or datafiles are taken offline, restored from backups, recovered and placed online. No changes are lost and the database remains available during the recovery.

If an archive log or online log that is required for recovery is also lost, then you should perform point-in-time database or tablespace recovery.

If one or more datafiles and/or all control files are lost, and the database is not open, then you cannot use this procedure. All lost files are restored from backups,

recovered and the database is opened. No changes are lost, but the database is unavailable during recovery. If an archive log or online log required for recovery is also lost, then you should perform point-in-time database recovery.

Has a User Accidentally Destroyed Data?

You should perform point-in-time database recovery when an archive log or online log required for recovery is lost, or to recover from user errors. All datafiles and the control file are restored from backups taken before the point-in-time, recovered to the point-in-time and the database is opened. All changes made subsequent to the point-in-time are lost. If the database remains open and no tablespaces that contain rollback segments are lost, you can also use point-in-time tablespace recovery.

Point-in-Time Tablespace Recovery All changes made to the recovered tablespaces after the point-in-time are lost. The database, excluding the recovered tablespaces, is available during recovery.

Have You Tested Backup and Recovery Strategies?

You should test your backup and recovery strategies in a test environment before moving to a production system. You should continue to test your system regularly. That way, you can test the thoroughness of your strategies and later avoid real-life crises. Performing test recoveries regularly ensures that your archiving and backup procedures work. It also keeps you familiar with recovery procedures, so that you are less likely to make mistakes in a crisis.

Recovery Operations

Media recovery restores a database's datafiles to a point-in-time before disk failure, and includes the committed data in memory that was lost due to failure. Following is a list of media recovery operations:

- Complete Media Recovery

  Complete media recovery includes the application of all necessary redo or incremental backups ever generated for the particular incarnation of the database being recovered. Complete media recovery can be performed on offline datafiles while the database is open. Complete media recovery may require an open resetlogs operation if a backup control file is included in the recovery, or a resetlogs operation creating a new control file has been performed. Types of complete media recovery are:

  – Closed Database Recovery

  – Open-Database, Offline-Tablespace Recovery

  – Open-Database, Offline-Tablespace, Individual Datafile Recovery

- Incomplete Media Recovery

  Incomplete media recovery produces a version of the database as it was at some time in the past. Incomplete media recovery must either continue on to become a complete media recovery, or be terminated by an open resetlogs operation that creates a new incarnation of the database. The database must be closed for incomplete media recovery operations. Types of incomplete media recovery are:

  – Time-based Recovery

    You specify the point in time to recover to

  – Cancel-based Recovery

    You decide to cancel during the recovery operation

  – Change-based Recovery

  For Recovery Manager, types of incomplete media recovery are:

  – Time-based Recovery

  – Change-based Recovery

  – Log Sequence Recovery (replaces cancel-based recovery, above)

    Recovers up to a specified log sequence number

5

Choosing a Backup Method

This chapter describes the different backup methods available and situations when each should be used, and contains the following topics:

- Backup Methods and Requirements
- Recovery Manager
- Feature Comparison of Backup Methods

Backup Methods and Requirements

This section describes the various Oracle backup methodologies and requirements associated with each, and includes the following topics:

- Table 5–1 identifies additional requirements associated with the different backup methods.
- Operating System (O/S)
- Enterprise Backup Utility (EBU)
- Export

Table 5–1 identifies additional requirements associated with the different backup methods.

Table 5–1 Requirements for Different Backup Methods

Backup Method	Version Available	Requirements
Recovery Manager	Oracle8	Media Manager (if backing up to tape)
O/S	All versions of Oracle	O/S backup utility (for example, UNIX dd)
Export	All versions of Oracle	N/A
Enterprise Backup Utility (EBU)	Oracle7	Media Manager (if backing up to tape)

Recovery Manager

The Recovery Manager utility manages the backup, restore and recovery operations of Oracle databases. Recovery Manager uses information about the database to automatically locate, then back up, restore and recover datafiles, control files and archived redo logs.

Operating System (O/S)

An O/S backup is performed via a native command on your O/S, which is used to make backups of Oracle database files. For example, you can use the ‘dd’ or ‘tar’ commands to create UNIX O/S backups. O/S backups are controlled by UNIX scripts written and maintained by the Database Administrator, and/or the System Administrator.

See Also: For information about the utilities available on your operating system, see your operating system-specific documentation.

Export

The Oracle Export utility writes data from an Oracle database to operating system files in an Oracle database format. Export files store information about schema objects created for a database. Because the Oracle Export utility can selectively export specific objects, you may consider exporting portions of or all of a database for supplemental protection and flexibility. Database exports are not a substitute for whole database backups and don't provide the same recovery advantages that the built-in functionality of Oracle offers.

See Also: For information about using exports to supplement your backup strategy, see *Oracle8 Utilities*.

Enterprise Backup Utility

The Oracle Enterprise Backup Utility is available to back up Oracle7 databases; it is not compatible with Oracle8 databases.

See Also: For information about using the EBU to back up an Oracle7 database, see the *Oracle7 Enterprise Backup Utility Administrator's Guide*.

Recovery Manager

The Recovery Manager utility manages the backup, restore and recovery operations of Oracle databases. Recovery Manager uses information about the database to automatically locate, then back up, restore and recover datafiles, control files and archived redo logs.

Recovery Manager gets the required information from either the databases' control file, or via a central repository of information called a recovery catalog, which is maintained by Recovery Manager.

Using Oracle Enterprise Manager to Perform Recovery Manager Backups

You can perform Recovery Manager backups using Oracle Enterprise Manager. Oracle Enterprise Manager-Backup Manager is a GUI interface to Recovery Manager that enables you to perform backup and recovery via a point-and-click method.

See Also: For information about performing backup and recovery using Oracle Enterprise Manager, see the *Oracle Enterprise Manager Administrator's Guide*.

Recovery Manager is Different from Traditional Operating System Backups

Recovery Manager is a command line interface (CLI) that directs an Oracle server process to back up, restore or recover the database it is connected to.

The Recovery Manager program issues commands to an Oracle server process.The Oracle server process reads the datafile, control file or archived redo log being backed up, or writes the datafile, control file or archived redo log being restored or recovered.

When an Oracle server process reads datafiles, it detects any split blocks and re-reads them to get a consistent block. Hence, you should not put tablespaces in hot backup mode when using Recovery Manager to perform open backups.

See Also: "Fractured Block Detection During Open Database Backups in Recovery Manager" on page 7-26.

Recovery Manager performs backup and recovery procedures, and greatly simplifies the tasks administrators perform during these processes. For example, Recovery Manager provides a way to:

- Configure frequently executed backup operations
- Generate a printable log of all backup and recovery actions
- Use the recovery catalog to automate both media restore and recovery operations
- Perform automatic parallelization of backups and restores
- Find datafiles that require a backup based on user-specified limits on the amount of redo that must be applied
- Back up the database, individual tablespaces or datafiles

Backups to Disk

You can use Recovery Manager to perform backups to disk. You don't need to integrate Oracle with any Media Managers if you only intend to write backups to disk.

Backups to Sequential Media

Attention: In order to utilize tape storage for your Oracle database backups, Recovery Manager requires you to install a media management product. The Oracle Backup Solutions Program (BSP) provides customers with a range of media management products that are compliant with the Oracle Media Management interface specification. One of these products is the Legato Storage Manager (LSM), which is available for a number of platforms, and is included in the Oracle software shipped for those platforms.

Refer to your platform-specific documentation for information regarding availability of the Legato product on your platform. The *Legato Storage Manager Administrator's Guide* describes the features supported by this product. If other BSP media management products are included with your shipment, they will be documented in your platform-specific documentation.

Several other products may be available for your platform directly from a media management vendor. For a current list of available products, refer to the Oracle Backup Solutions Program (BSP) web site at:

http://www.oracle.com/st/products/features/backup.html

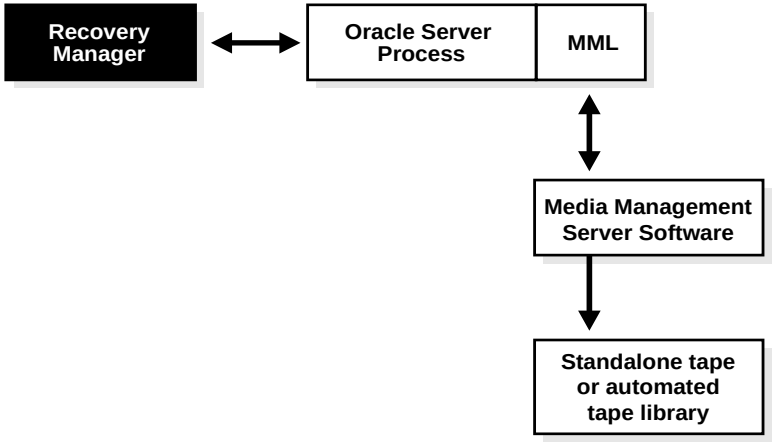
You can also contact your Oracle representative for a complete list.

If you wish to use a particular media management product, contact the media management vendor directly to determine if they are a member of the Oracle Backup Solutions Program.

To make backups to sequential media (such as tape) using Recovery Manager, you must have media management software integrated with the Oracle software.

Figure 5–1 shows the architecture for media management software integrated with Oracle.

Figure 5–1 Architecture for MML Integrated with Oracle



The Oracle executable is the same one started when a user connects to the database. The MML (Media Management Library) in the diagram above represents Media Management vendor-supplied software that can interface with Oracle. Oracle calls MML software routines to back up and restore datafiles to and from media controlled by the Media Manager.

The following Recovery Manager script performs a datafile backup to a tape drive controlled by a Media Manager:

```
run {
   allocate channel t1 type 'SBT_TAPE';
   backup
      format 'df_%s_%t'
      (datafile 10);
}
```

When Recovery Manager executes the above command, it sends the backup request to the Oracle server performing the backup. The Oracle server process identifies the output channel as the type 'SBT_TAPE', and requests the Media Management Library to load a tape and write the output specified.

When Oracle requests the restore of a particular file, the Media Manager identifies the tape the file is on, reads the tape, and passes the information back to the Oracle server process, which then writes the file to disk.

The Media Manager labels and keeps track of the tape and names of files on each tape. If your site owns an Automated Tape library with robotic arm, the Media

Manager will automatically load and unload the tapes required by Oracle; if not, the Media Manager will request an operator to load a specified tape into the drive.

The Oracle server doesn't need to connect to the MML software when a backup is written to disk. The Oracle executable writes backups to disk without third party integration software.

Feature Comparison of Backup Methods

Table 5–2 compares the features of the backup methods described in this chapter

Table 5–2 Feature Comparison of Backup Methods

Feature	Recovery Manager	Operating System	Export
closed database backups	Supported. Requires instance to be mounted.	Supported.	Not supported.
open database backups	You do not use BEGIN / END BACKUP commands.	Generates more redo when using BEGIN / END BACKUP commands.	Requires RBS to generate consistent backups.
incremental backups	Supported. Backs up all modified blocks.	Not supported.	Supported, but not a true incremental, as it backs up a whole table even if only one block is modified.
corrupt block detection	Supported. Identifies corrupt blocks and writes to V$BACKUP_CORRUPTION or V$COPY_CORRUPTION.	Not supported.	Supported. Identifies corrupt blocks in the export log.
automatically backs up data	Supported. Establishes the name and locations of all files to be backed up (whole database, tablespace, datafile or control file backup).	Not supported. Files to be backed up must be specified manually.	Supported. Performs either full, user or table backups.

Table 5–2 Feature Comparison of Backup Methods (Cont.)

Feature	Recovery Manager	Operating System	Export
catalogs backups performed	Supported. Backups are cataloged to the recovery catalog and to the control file, or just to the control file.	Not supported.	Not supported.
makes backups to tape	Supported. Interfaces with a Media Manager.	Supported. Backup to tape is manual or managed by a Media Manager.	Supported.
Backs up init.ora and password files	Not supported.	Supported.	Not supported.
Operating System Independent language	An O/S independent scripting language.	O/S dependent.	O/S independent scripting language.

Part II

Backup and Recovery—Recovery Manager

6

Getting Started with Recovery Manager

This chapter is a “quick start” to using Recovery Manager, and includes the following topics:

- Decisions to Make Before Using Recovery Manager
- Recovery Manager Connection Options
- Running Recovery Manager Commands
- Recovery Manager Sample Scripts and Scenarios
- Prerequisites for Performing Backups to Tape

Decisions to Make Before Using Recovery Manager

You will need to make the following decisions before you start using Recovery Manager:

- Will You Use a Recovery Catalog?
- Decide Whether or Not to Use Password Files
- Decide How to Back Up init.ora Files and Password Files

Will You Use a Recovery Catalog?

There are costs and benefits associated with using and not using a Recovery Catalog. Following is a brief overview of some of these costs and benefits.

Costs and Benefits When Using a Recovery Catalog

When you use a recovery catalog, Recovery Manager can perform a wider variety of automated backup and recovery functions; however, Recovery Manager requires that you maintain a recovery catalog schema, and any associated space used by that schema.

If you use a recovery catalog, you must decide which database you will use to install the recovery catalog schema, and also how you will back this database up.

The size of the recovery catalog schema:

- is dependent on the number of databases monitored by the catalog
- is dependent on the number and size of Recovery Manager scripts stored in the catalog
- will grow as the numbers of archived logs and backups for each database grow

If you use Recovery Manager to back up many databases, you may wish to create a separate recovery catalog database, and create the Recovery Manager user in that database. You should also decide whether or not to operate this database in ARCHIVELOG mode.

If you store the recovery catalog in a separate database, you will require a small amount of disk space for the following:

- system
- temp
- rollback segment tablespaces
- online redo log

An additional benefit of maintaining a separate recovery catalog is that it is only unavailable at the DBA's discretion. Most of the space used in this database is devoted to supporting tablespaces (for example, system, temp, and rollback).

Table 6–1 lists typical space requirements for 1 year.

Table 6–1 Typical Recovery Catalog Space Requirements for 1 Year

Type of Space	Space Requirement
system	50 megabytes
temp	5 megabytes
rollback	5 megabytes
recovery catalog	10 megabytes
online logs	1 megabyte each (3 groups, each with 2 members)

If you have more than one database to back up, you can create more than one recovery catalog and have each database serve as the other's recovery catalog. For example, assume there are two production databases, one called "prd1," and a second called "prd2." You can install the recovery catalog for "prd1" in the "prd2" database, and the recovery catalog for the "prd2" database in "prd1." This enables you to avoid the extra space requirements and memory overhead of maintaining a separate recovery catalog database. However, this solution is not practical if the recovery catalog databases for both reside in tablespaces residing on the same physical disk.

Note: You *must* install the recovery catalog schema in a different database from the target database you will be backing up. If you don't, the benefits of using a recovery catalog are lost if you lose the database and need to restore.

WARNING: Ensure that the recovery catalog and target databases do *not* reside on the same disks. If they are on the same disks and you lose one, you will most likely lose the other.

See Also: For more information, see Chapter 7, "Recovery Manager Concepts".

Costs and Benefits When not Using a Recovery Catalog

The following recovery features are not available when you do not use a recovery catalog:

- tablespace point-in-time recovery
- stored scripts
- recovery when the control file is lost or damaged

In order to restore and recover your database without using a recovery catalog, Oracle recommends that you:

- Use multiplexed control files (minimum of two), and have each control file on separate disks to protect against media failure.
- Keep excellent record of what files were backed up, the date they were backed up, and also the names of the backup pieces each file was written to. Keep all Recovery Manager backup logs.

WARNING: It is difficult to restore and recover if you lose your control files and do not use a recovery catalog. The only way to restore and recover when you have lost all control files and need to restore and recover datafiles, is to call Oracle WorldWide Customer Support (WWCS). WWCS will need to know the following:

- **current schema of the database**
- **which files were backed up**
- **what time the files were backed up**
- **names of the backup pieces containing the files**

Also, Oracle recommends using Recovery Manager with a recovery catalog.

Setting Up the Recovery Catalog Schema

When you use a recovery catalog, you need to set up the schema. Oracle suggests you put the recovery catalog schema in its own tablespace; however, it could be put in the system tablespace, if necessary.

To Set Up the Recovery Catalog Schema

1. Using Server Manager (LineMode) connect internal (or as SYSDBA) to the database containing the recovery catalog.
2. Issue the following commands:

```
spool create_rman.log
connect internal
create user rman identified by rman
temporary tablespace temp
default tablespace rcvcat quota unlimited on rcvcat;
grant recovery_catalog_owner to rman;
connect rman/rman
@?/rdbms/admin/catrman
```

3. Check the create_rman.log file for any errors before continuing.

Assume the following for the recovery catalog schema created here:

- there is a tablespace called “rcvcat” that stores the recovery catalog
- there is a tablespace called “temp”
- the database is configured in the same way as all “normal” databases (for example, catalog.sql and catproc.sql have successfully run)

> **Note:** You must not run catrman.sql script in the SYS schema. Run the catrman.sql script in the recovery catalog schema.

Also, you *must* install the recovery catalog schema in a different database from the target database you will be backing up. If you don’t, the benefits of using a recovery catalog are lost if you lose the database and need to restore.

> **WARNING: Ensure that the recovery catalog and target databases do *not* reside on the same disks; if they do and you lose one, you will most likely lose the other.**

Decide When and How to Back Up the Recovery Catalog

It is important that you maintain regular backups of your recovery catalog—Oracle recommends a daily backup.

You should also use 2 different methods to back up your Recovery Catalog. For example, you could use Recovery Manager *and* the Export utility, or Recovery Manager *and* an operating system utility.

If you use Recovery Manager to back up the recovery catalog database, create a second recovery catalog in a separate database (such as the production database you are backing up). If you use the recovery catalog to back itself up, and lose this database due to media failure, you won't easily be able to restore it.

Decide How to Resynchronize the Recovery Catalog

You should automate recovery catalog resynchronizations so that a resychronization is performed at least once a day. If your site generates many archived logs each day, then you should resynchronize the catalog once an hour.

It typically takes a short amount of time to resynchronize the catalog, and the shorter the time lapse between resynchronizations, the faster the resynchronization process will be.

At least once a week verify that the automatic catalog resynchronization process you set up, works.

> **Note:** You should manually resynchronize the catalog whenever you make any structural changes to the database (such as adding a file, or dropping a tablespace).

Decide Whether or Not to Use Password Files

Generally, you need to use a password file when connecting to the target database over a non-secure Net8 connection, especially when you do the following:

- You run Recovery Manager remotely (from a different machine than the database you wish to back up). For example, if you choose to use Oracle Enterprise Manager— Backup Manager GUI.
- You wish to use Recovery Manager, and use a TNS alias in the connect string for the target database.
- You have a Parallel Server database, and wish to backup this database from more than one node in the cluster concurrently, while using only one Recovery Manager session.

See Also: *Oracle8 Parallel Server Concepts and Administration*, for an example of a backup distributed over two nodes in an OPS cluster.

Decide How to Back Up init.ora Files and Password Files

Recovery Manager does not back up init.ora and password files. You must plan how you will protect these files from media failure.

Recovery Manager Connection Options

This section includes the following sample Recovery Manager connection options:

- Connecting to Recovery Manager Without a Recovery Catalog
- Connecting to Recovery Manager With a Recovery Catalog

> **Note:** The examples in this section are not used with Oracle Enterprise Manager Backup Manager GUI. Also, there are many more Recovery Manager connection options than shown in the examples here.

To Perform a General Connect to Recovery Manager The following is the general form of the Recovery Manager connect string:

```
rman target internal/<pwd>@prd1 rcvcat rman/rman@rcat
```

Connecting to Recovery Manager Without a Recovery Catalog

In this example, assume that:

- the target database is called “prod1” (and has the same TNS alias)
- “scott” has been granted SYSDBA privileges (password is “tiger”)

If the target database does not have a password file, the user you are logged in as must be validated using O/S authentication (similar to svrmgrl, and connect internal).

To Connect to Recovery Manager Without Password Files Set your ORACLE_SID to be the target database, and issue the following statement:

```
rman nocatalog
RMAN> connect target
```

To Connect to Recovery Manager With Password Files If the target database uses password files, you can connect using:

```
rman target internal/kernel@prod1 nocatalog
```

or

```
rman nocatalog
RMAN> connect target internal/kernel@prod1
```

Recovery Manager automatically requests a connection to the target database as SYSDBA.

If you wish to connect as a DBA user who has SYSDBA privileges, issue the following statement:

```
rman target scott/tiger@prod1 nocatalog
```

Connecting to Recovery Manager With a Recovery Catalog

In this example, assume that:

- the recovery catalog database is called “rcat” (and has the same TNS alias)
- the schema containing the recovery catalog is “rman” (password is “rman”)
- the target database is called “prod1” (and has the same TNS alias)
- “scott” has been granted SYSDBA privileges (password is “tiger”)

To Perform a Connect Using Password Files In the following statement the administrator is connecting using password files for the target database, and the password to connect internal is ‘kernel’.

```
rman target internal/kernel@prod1 rcvcat rman/rman@rcat
```

To Perform a Connect Without Using Password Files In this statement Recovery Manager is running on the same machine as the target database. The administrator starts Recovery Manager by setting the ORACLE_SID to the target instance, then issuing the following statement:

```
rman rcvcat rman/rman@rcat
RMAN> connect target
```

To Connect as a Specific User with SYSDBA Here, Recovery Manager automatically requests a connection to the target database as SYSDBA using O/S authentication, not a password file:

```
rman target scott/tiger@prod1 rcvcat rman/rman@rcat
```

Running Recovery Manager Commands

This section includes the following topics:

- Running Recovery Manager Commands Interactively: Example 1
- Using Command Files: Example 2
- Using Stored Scripts: Example 3
- Specifying Time Parameters in Recovery Manager

Catalog maintenance commands are executed outside of a **run** command as follows:

```
rman target internal/kernel@prod1 rcvcat rman/rman@rcat
RMAN> resync catalog;
```

Commands such as **backup**, **restore**, and **allocate** must be executed within a **run** command as follows:

```
rman target internal/kernel@prod1 rcvcat rman/rman@rcat
RMAN> run {
2> allocate channel c1 type disk;
3> copy datafile 5 to '/dev/backup/prod1/prod1_tab7.dbf';
4> }
```

Running Recovery Manager Commands Interactively: Example 1

To run Recovery Manager commands interactively, you need to start Recovery Manager, then type the commands into the command line interface, as follows:

```
rman target internal/kernel@prod1 rcvcat rman/rman@rcat
RMAN> register database;
RMAN> run {
2>   allocate channel d1 type disk;
3>   allocate channel d2 type disk;
4>   allocate channel d3 type disk;
5>   backup
6>     incremental level 0
7>     tag db_level_0
8>     filesperset 6
9>     format '/dev/backup/prod1/df/df_t%t_s%s_p%p'
10>     (database);
11>    sql 'alter system archive log current';
12>    backup
13>     filesperset 20
14>     format '/dev/backup/prod1/al/al_t%t_s%s_p%p'
15>     (archivelog all
16>      delete input);
17> }
```

Using Command Files: Example 2

You can type Recovery Manager commands into a file, and then run the command file by specifying its name on the command line.

The contents of the command file should be identical to commands entered at the command line.

Here, the Recovery Manager script from Example 1 has been placed into a file (called" b_whole_l0.rcv"). This file is run from the command line as follows.

```
rman target internal/kernel@prod1 rcvcat rman/rman@rcat cmdfile b_whole_l0.rcv
```

Using Stored Scripts: Example 3

A *stored script* is a set of Recovery Manager commands (enclosed in braces) that are stored in the recovery catalog. A stored script only relates to one database. You cannot use stored scripts if you do not use a recovery catalog.

To create a stored script, you can either enter your script interactively into the Recovery Manager command line interface (as in Example 1), or type your Recovery Manager commands into a command file, and run the command file (as in Example 2).

Following is a stored script:

```
replace script b_whole_l0 {
# Backup Whole database, and archived logs
  allocate channel d1 type disk;
  allocate channel d2 type disk;
  allocate channel d3 type disk;
  backup
    incremental level 0
    tag b_whole_l0
    filesperset 6
    format '/dev/backup/prod1/df/df_t%t_s%s_p%p'
     (database);
    sql 'alter system archive log current';
    backup
     filesperset 20
     format '/dev/backup/prod1/al/al_t%t_s%s_p%p'
     (archivelog all
      delete input);
}
```

Specifying Time Parameters in Recovery Manager

Before invoking Recovery Manager, set the NLS_DATE_FORMAT and NLS_LANG environment variables. These variables specify the format used for the time parameters in Recovery Manager commands, such as:

- **restore** and **recover**
- **report**

The following example shows typical environment variables:

```
NLS_LANG=american
NLS_DATE_FORMAT='Mon DD YYYY HH24:MI:SS'
```

> **Note:** Both NLS_LANG and NLS_DATE_FORMAT must be set for NLS_DATE_FORMAT to be used.

Recovery Manager Sample Scripts and Scenarios

There are a number of sample scripts in the RDBMS demo directory. These files are executable Recovery Manager command files and are fully commented to explain the features used. You can edit them to customize them for your site.

The file `case1.rcv` creates a number of stored scripts that back up, restore and recover a database. These scripts are typical of how some DBAs back up their databases, and you can use them as a starting point for developing backup, restore and recovery scripts.

The following files contain either a backup or recovery scenario. These scripts show more of the Recovery Manager syntax flexibility and power than appears in the sample scripts:

```
case2.rcv - incomplete recovery
case3.rcv - disaster recovery
case4.rcv - consistent backup
```

See Also: Additional backup and recovery scenarios are in Chapter 9, "Recovery Manager Scenarios".

Prerequisites for Performing Backups to Tape

To back up to and restore from sequential media, such as tape, you must have a Media Manager integrated with Oracle.

This section includes the following topics:

- Linking with a Media Manager
- Generating Unique File Names
- Know Your Media Manager's Maximum File Size Limit

Linking with a Media Manager

To integrate Oracle with a Media Manager, you must:

- have Media Manager software and hardware installed and configured
- obtain that Media Management Vendor's library interface software (Media Management Library) from that Vendor

The Media Management Library (MML) is then linked with the Oracle kernel software. This enables Oracle server processes to call the Media Manager.

> **Note:** For instructions on how to achieve this on your platform, see your operating system-specific Oracle documentation and the documentation supplied by your Media Manager.

Generating Unique File Names

You should use the substitution variables provided by Recovery Manager to generate unique backup piece names when writing backups to a Media Manager. A backup piece name is determined by the format string specified either in the **backup** command, or in the **allocate channel** command.

The Media Manager considers the backup piece name as the 'file name backed up', so this name must be unique in the Media Manager catalog.

> **WARNING:** When a backup generates a file whose name already exists in the Media Manager's catalog, the original file having that name is deleted.

You can use the following substitution variables to make unique format strings:

%d - database name (uppercase)

%t - backup set stamp

%s - backup set number

%p - piece number in the set

%u - an 8 character id composed of compressed representations of the backup set number and the time the backup set was created

Note: Some Media Managers only support a 14-character backup piece name. See your Media Management documentation to determine the limit for your Media Manager.

Know Your Media Manager's Maximum File Size Limit

Some Media Managers have limits on the maximum size of files they can back up or restore.

As Recovery Manager multiplexes multiple datafiles into one output file (one backup piece), it is possible that the piece size is in excess of the size that many Media Managers or file systems are able to store.

To avoid problems, find out what operational limits on file sizes exist, and ensure the files written out by Recovery Manager do not exceed these limits.To limit backup piece file sizes, use the keyword **kbytes** in a **setlimit channel** command (see scenario `case1.rcv` in the demo directory for an example).

7

Recovery Manager Concepts

This chapter describes the basic concepts for the Oracle Recovery Manager utility, and includes the following topics:

- Introduction to Recovery Manager
- Backing Up to Sequential Media
- Recovery Manager Backup Types
- Corruption Detection
- Parallelization
- Report Generation
- User Tags for Recovery Manager Backups
- Backup Constraints
- Restore Constraints
- Integrity Checking
- Fractured Block Detection During Open Database Backups in Recovery Manager
- Cataloging Image Copies and Archive Logs

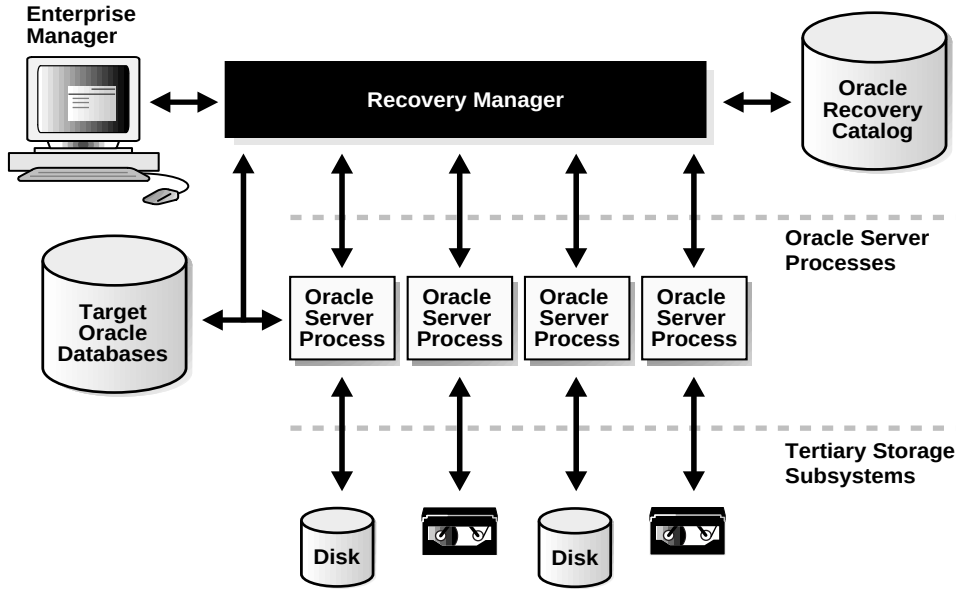
Introduction to Recovery Manager

Recovery Manager is an Oracle utility you use to back up, restore, and recover database files. Recovery Manager starts Oracle server processes on the database to be backed up or restored (called the *target database*). These Oracle server processes actually perform the backup and restore. For example, during a backup, the server process reads the files to be backed up, and writes out the files out to your target storage device.

Recovery Manager performs important backup and recovery procedures, and greatly simplifies the tasks administrators perform during these processes. For example, Recovery Manager provides a way to:

- Configure frequently executed backup operations
- Generate a printable log of all backup and recovery actions
- Use the recovery catalog to automate both restore and recovery operations
- Perform automatic parallelization of backups and restores
- Find datafiles that require a backup based on user-specified limits on the amount of redo that must be applied
- Back up the whole database, selected tablespaces, or selected datafiles

Figure 7–1 Recovery Manager



Recovery Manager has a command language interpreter (CLI), and can be executed in interactive or batch mode. You may also specify a log file on the command line to record significant Recovery Manager actions.You can also use Recovery Manager via the Enterprise Manager Recovery Manager application.

When running in batch mode, Recovery Manager reads input from a command file and writes output messages to a log file (if specified). The command file is parsed in its entirety before any commands are compiled and executed. Batch mode is most suitable for performing regularly scheduled backups via an operating system job control facility.

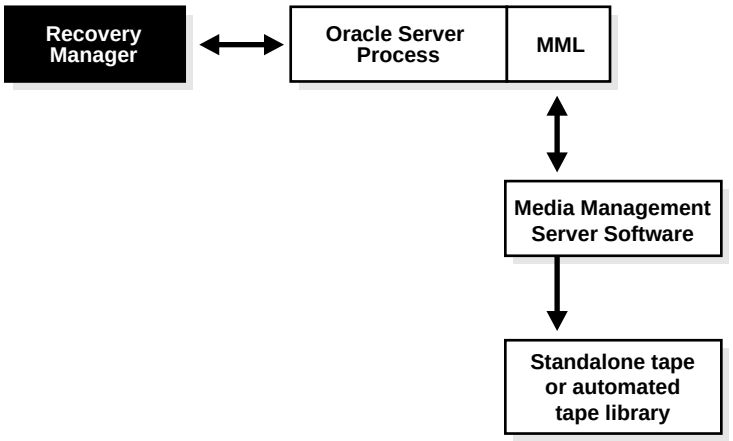
Recovery Manager provides the following categories of commands:

- **backup**, **restore**, **copy** and **recover**
- Recovery catalog maintenance commands
- Stored script maintenance commands
- **report** and **list** commands

Backing Up to Sequential Media

If you use Recovery Manager to perform backups to sequential media, such as tape, you must have media management software integrated with your Oracle software.

Figure 7–2 MML Architecture



In Figure 7–2, the Oracle executable is the same executable that is started when a user connects to the database. *MML* is the Media Management Library, which is media management, vendor-supplied software. Software routines in the MML are called by Oracle to back up and restore.

In the following example, a Recovery Manager script performs a datafile backup:

```
run {
   allocate channel t1 type 'SBT_TAPE';
   backup
     format 'df_%s_%t'
  (datafile 10);
}
```

When Recovery Manager executes this command, it sends the backup request to the Oracle server process that will be performing the backup. The Oracle server process identifies the output channel as the type 'SBT_TAPE', and requests the MML to load the tape and write the output, among other things.

If the channel type was disk, Oracle would not have connected to the MML software because Oracle can write to disk without the assistance of third-party integration software.

See Also: For information about vendors offering software integrated with Recovery Manager, see your Oracle representative.

Backing Up Using Enterprise Manager

You can perform Recovery Manager backups using Oracle Enterprise Manager-Backup Manager software, which is a graphical user interface to Recovery Manager.

See Also: See your operating system-specific documentation for more information.

The Recovery Catalog

The *recovery catalog* is a repository of information that is used and maintained by Recovery Manager. Recovery Manager uses the information in the recovery catalog to determine how to execute requested backup and restore actions.

The recovery catalog contains informations about the following:

- datafile and archivelog backup sets and backup pieces
- datafile copies
- archived redo logs and copies of them
- tablespaces and datafiles at the target database
- named user-created sequences of Recovery Manager and SQL commands called *stored scripts*

Operating with a Recovery Catalog

Oracle recommends you use Recovery Manager with a recovery catalog, especially if you have 20 (or more) datafiles. You must frequently resynchronize the recovery catalog with the target database control file to keep it up to date. The more up-to-date the recovery catalog is, the easier it will be to recover. For example, if you have 25 archive logs that have been created subsequent to the last resynchronization, and you experience a failure requiring that you restore the entire database and control file, you must first catalog these files with the recovery catalog before you can use them.

The recovery catalog is maintained solely by Recovery Manager, and the target database never accesses it directly. Recovery Manager propagates information

about the database structure, archived redo logs, backup sets and datafile copies into the recovery catalog from the target database's control file.

You do not need an additional database which is solely for the recovery catalog. You can put the recovery catalog in an existing database. It is the database administrator's responsibility to make the recovery catalog database available to Recovery Manager. Database administrators are also responsible for taking backups of the recovery catalog. Since the recovery catalog resides in an Oracle database, administrators can use Recovery Manager to back it up by reversing the roles of the recovery catalog database and the target database. In other words, the target database can become the recovery catalog database and the recovery catalog database can be treated as a target database.

A single recovery catalog is able to store information for multiple target databases.

If the recovery catalog is destroyed and no backups are available, then you can partially reconstruct the catalog from the current control file or control file backups. However, you should always aim to have a valid, recent backup of your recovery catalog.

Propagating Information from the Control File

The size of the target database's control file will grow, depending on the number of:

- backups performed
- archive logs created
- days (minimum number) this information is stored in the control file

You can specify the minimum number of days this information is kept in the control file using the parameter CONTROL_FILE_RECORD_KEEP_TIME. Entries older than the number of days are candidates for overwrites by newer information. The larger the CONTROL_FILE_RECORD_KEEP_TIME setting is, the larger the control file will be.

At a minimum, you should resynchronize your recovery catalog at intervals less than the CONTROL_FILE_RECORD_KEEP_TIME setting, because after this number of days, the information in the control file will be overwritten with the most recently created information; if you have not resynchronized, and information has been overwritten, this information can not be propagated to the recovery catalog.

> **Note:** The maximum size of the control file is port specific. See your see your operating system-specific Oracle documentation.

See Also: For more information about the CONTROL_FILE_RECORD_KEEP_TIME parameter, see the *Oracle8 Reference*.

Operating without a Recovery Catalog

Note: You are not required to maintain a recovery catalog with Recovery Manager; however, Oracle recommends that you use one.

Because most information in the recovery catalog is also available in the target database's control file, Recovery Manager supports an operational mode where it uses the target database control file instead of a recovery catalog. This operational mode is appropriate for small databases where installation and administration of another database for the sole purpose of maintaining the recovery catalog would be burdensome.

Note that the following features are not supported in this operational mode:

- tablespace point-in-time recovery
- stored scripts
- *restore and recovery* when the control file is lost or damaged

To restore and recover your database without using a recovery catalog, Oracle recommends that you:

- Use multiplexed control files (minimum of two), and have each control file on separate disks to protect against media failure.

- Keep excellent record of what files were backed up, the date they were backed up, and also the names of the backup pieces each file was written to. Keep all Recovery Manager backup logs.

  WARNING: It is difficult to restore and recover if you lose your control files and do not use a recovery catalog. The only way to restore and recover when you have lost all control files and need to restore and recover datafiles, is to call Oracle WorldWide Support (WWS). WWS will need to know the following:

  - **current schema of the database**
  - **which files were backed up**
  - **what time the files were backed up**
  - **names of the backup pieces containing the files**

Keeping Information in the Control File

The size of the target database's control file will grow, depending on the number of:

- backups performed
- archive logs created
- days (minimum) this information is stored in the control file

You can specify the minimum number of days this information is kept in the control file using the parameter CONTROL_FILE_RECORD_KEEP_TIME. Entries older than this number of days are candidates to be overwritten by newer information. The larger the CONTROL_FILE_RECORD_KEEP_TIME is, the larger the control file will be.

If you do not use a recovery catalog, you may wish to set the CONTROL_FILE_RECORD_KEEP_TIME parameter to the duration (in days) you intend to keep your backups. This way, the backup information in the control file will not be overwritten before the backups become obsolete.

Note: The maximum size of the control file is port specific. See your see your operating system-specific Oracle documentation.

See Also: For more information about the CONTROL_FILE_RECORD_KEEP_TIME parameter, see the *Oracle8 Reference*.

Snapshot Control File

Recovery Manager generates a *snapshot control file*, which is a temporary backup control file each time it refreshes the recovery catalog. This snapshot control file ensures that Recovery Manager has a consistent view of the control file either when refreshing the recovery catalog or when querying the control file (if you are not using a recovery catalog). Because the snapshot control file is for Recovery Manager's short term use, it is not registered in the recovery catalog. Recovery Manager records the snapshot control file checkpoint in the recovery catalog to indicate precisely how current the recovery catalog is.

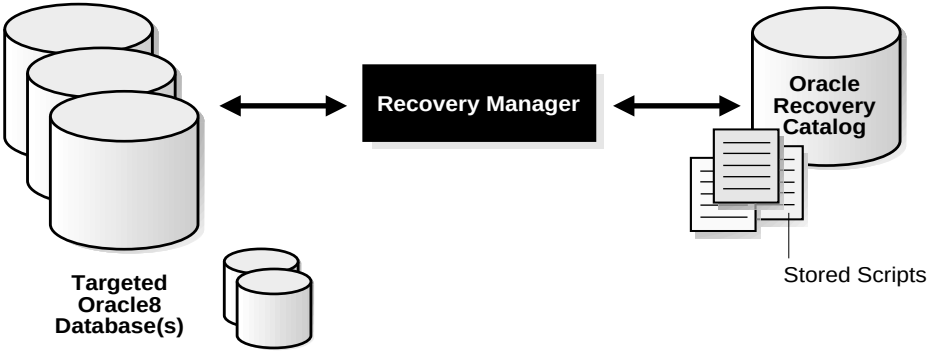
The Oracle8 server ensures only one Recovery Manager session accesses a snapshot control file at any point in time. This is necessary to ensure that two Recovery Manager sessions do not interfere with each other's use of the snapshot control file.

> **Note:** It is possible to specify the name and location of a snapshot control file. For information on how to do this, see "Configuring the Snapshot Control File Location" on page 8-19.

Stored Scripts

Database administrators can store a sequence of Recovery Manager commands, called a *stored script*, in the recovery catalog for execution at a later time. This allows the administrator to plan, develop and test a set of commands for backing up, restoring, and recovering the database. Stored scripts also minimize the potential for operator errors. Each stored script relates to only one database.

Figure 7–3 Recovery Manager Stored Scripts



Recovery Manager Backup Types

Recovery Manager supports two basic types of backups:

- Backup Sets
- Image Copies

Backup Sets

A *backup set* contains one or more datafiles or archivelogs; however, it cannot contain both datafiles and archivelogs. Datafile backup sets can also contain a control file backup.

Backup sets are in an Oracle proprietary format (similar to Export files). You must perform a restore operation to extract a file from a backup set.

A backup set is also complete set of backup pieces that constitute a full or incremental backup of the objects specified in the **backup** command.

The pieces of a backup set are written serially. (Striping a backup set across multiple output devices is not supported.) If multiple output devices are available, you can partition your backups so that multiple backup sets are created in parallel. Recovery Manager performs this backup partitioning automatically, if desired.

A backup set can:

- be written to disk or tertiary storage
- be full or incremental
- span multiple O/S files (each file is a piece of the set—typically, one set is comprised of only one piece)

You can write backup sets to either normal operating system disk files or to sequential output media (for example, magnetic tape) using a sequential output device or media management system that is available and supported by Oracle on your operating system. To determine which device types are supported by your operating system or Media Manager, query the V$BACKUP_DEVICE view.

A backup set containing archived logs is called an *archivelog backup set*. You cannot archive directly to tape. However, Recovery Manager allows you to back up archived logs from disk to tape. Also, during recovery Recovery Manager automatically stages the required archived logs from tape to disk.

See Also: For details about V$BACKUP_DEVICE, see the *Oracle8 Reference*.

Backup Pieces

A backup set is composed of one or more *backup pieces*, each piece being a single output file. You should restrict the size of a backup piece to whatever is the maximum file size supported by your Media Manager or O/S (if writing to disk). If you do not restrict this size, a backup set will be comprised of only one file.

Each backup piece contains control and checksum information that allows the Oracle server process to validate the correctness of the backup piece during a restore.

Backup Set Compression

Datafile blocks that have never been used are not written out to backup sets. Image copy backups of a datafile always contain all datafile blocks.

File Multiplexing

Datafile blocks included in the same backup set are multiplexed together. This means that the blocks from all of the files in the set are interspersed with all the other blocks.

You can control the number of datafiles that are backed up concurrently to a backup set using the **filesperset** parameter. Controlling concurrency is helpful if you want to keep a tape device streaming without saturating a single datafile with

too many read requests (which can subsequently degrade online performance). You can limit the read rate by using the **set limit readrate channel** command.

See Also: For a detailed description, see "Multiplexed Backup Sets" on page 7-22.

Full and Incremental Backup Sets

Datafile backup sets can be full or incremental. A full backup is a backup of one or more datafiles that contain all blocks of the datafile(s). An incremental backup is a backup of one or more datafiles that contain only those blocks that have been modified since a previous backup. These concepts are described in more detail in the following sections.

Full Backup Sets

A *full backup* copies all blocks into the backup set, skipping only datafile blocks that have never been used. No blocks are skipped when backing up archivelogs or control files.

A full backup is *not the same* as a whole database backup; *full* is an indicator that the backup is not incremental.

Also, a full backup has no effect on subsequent incremental backups, and is not considered part of the incremental strategy (in other words, a full backup does not affect which blocks are included in subsequent incremental backups).

Oracle allows you to create and restore full backups of the following:

- datafile
- datafile copy
- tablespace
- control file (current or backup)
- database

Archivelog backup sets are always full backups.

Incremental Backup Sets

An *incremental backup* is a backup of one or more datafiles that contain only those blocks that have been modified since a previous backup at the same or lower level; unused blocks are not written out.

A control file may be included in an incremental backup set; however, it is always included in its entirety (in other words, the full control file is always included; no blocks are skipped).

Oracle allows you to create and restore incremental backups of the following:

- datafile
- tablespace
- database

The *multi-level incremental backup* feature allows you to create different levels of incremental backups. Each level is denoted by an integer. By default, an incremental backup at any particular level consists of those blocks that have been modified since the last backup *at that level or lower.*

A level 0 incremental backup copies all blocks containing data; a level 0 incremental backup is the base for subsequent incremental backups.

An incremental backup at a level greater than 0 copies only those blocks that have changed since a previous incremental backup at the same or lower level. Hence, the benefit of performing incremental backups is that you don't back up all of the blocks all of the time. Incremental backups at levels greater than 0 only copy blocks that were modified. This means the backup size may be significantly smaller, and require much less time. The size of the backup file depends solely upon the number of blocks modified, as well as the incremental backup level chosen by the DBA.

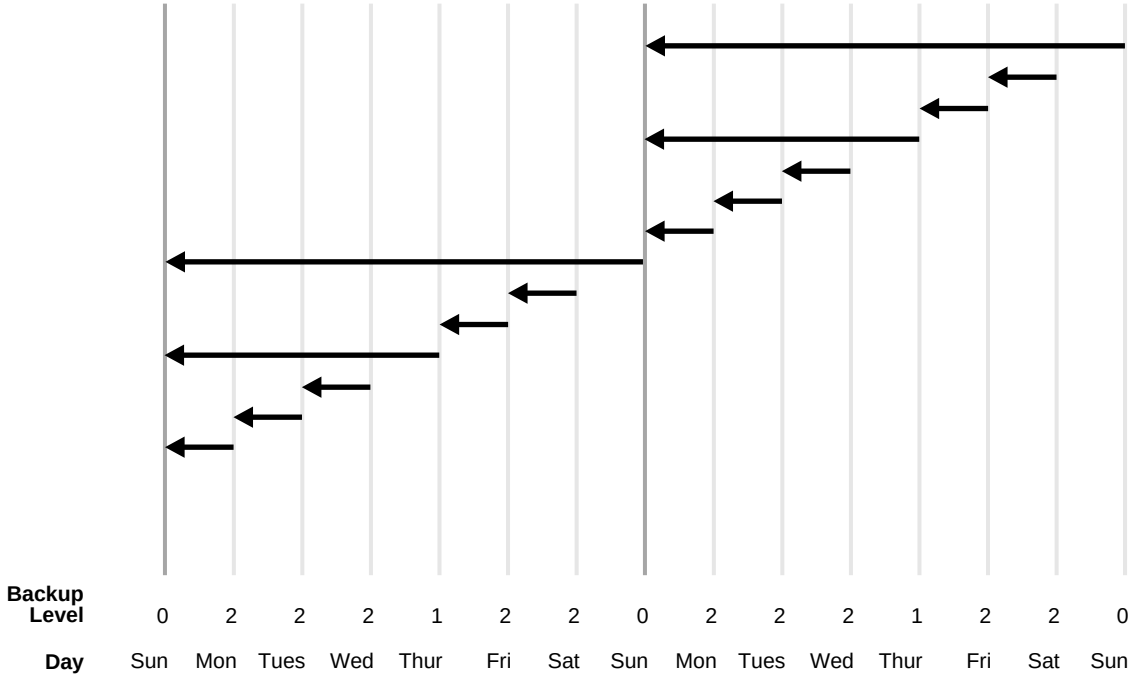
The multi-level incremental backup feature allows you to devise a backup scheme that enables you to restore the database within a given time constraint. For example, you could implement a three-level backup scheme so that a level 0 backup was a monthly full backup; a level 1 backup was a weekly incremental, and a level 2 was a daily incremental.

Cumulative Incremental Backup Sets

Oracle provides an option to make *cumulative* incremental backups at level 1 or greater. Cumulative incremental backups reduce the work needed for a restore by ensuring that you only need one incremental backup from any particular level at restore time. However, cumulative backups require more space and time because they duplicate the work done by previous backups at the same level.

A cumulative incremental backup copies all blocks that have changed since a previous incremental backup at a lower level.

Figure 7–4 Non-cumulative Incremental Backups(default type of incremental)



In the example above:

- Sunday

  An incremental level 0 backup occurs. This backs up *all* blocks that have ever been in use in this database.

- Monday

  An incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n or less; in this case the most recent incremental backup at level 2 or less is the level 0 Sunday backup, so only the blocks changed since Sunday will be backed up.

- Tuesday

  An incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n or less; in this case the most recent incremental backup at level 2 or less is the level 2 Monday backup, so only the blocks changed since Monday will be backed up.

- Wednesday

  An incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n or less; in this case the most recent incremental backup at level 2 or less is the level 2 Tuesday backup, so only the blocks changed since Tuesday will be backed up.

- Thursday

  An incremental level 1 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n or less; in this case the most recent incremental backup at level 1 or less is the level 0 Sunday backup, so only the blocks changed since the Sunday level 0 backup will be backed up.

- Friday

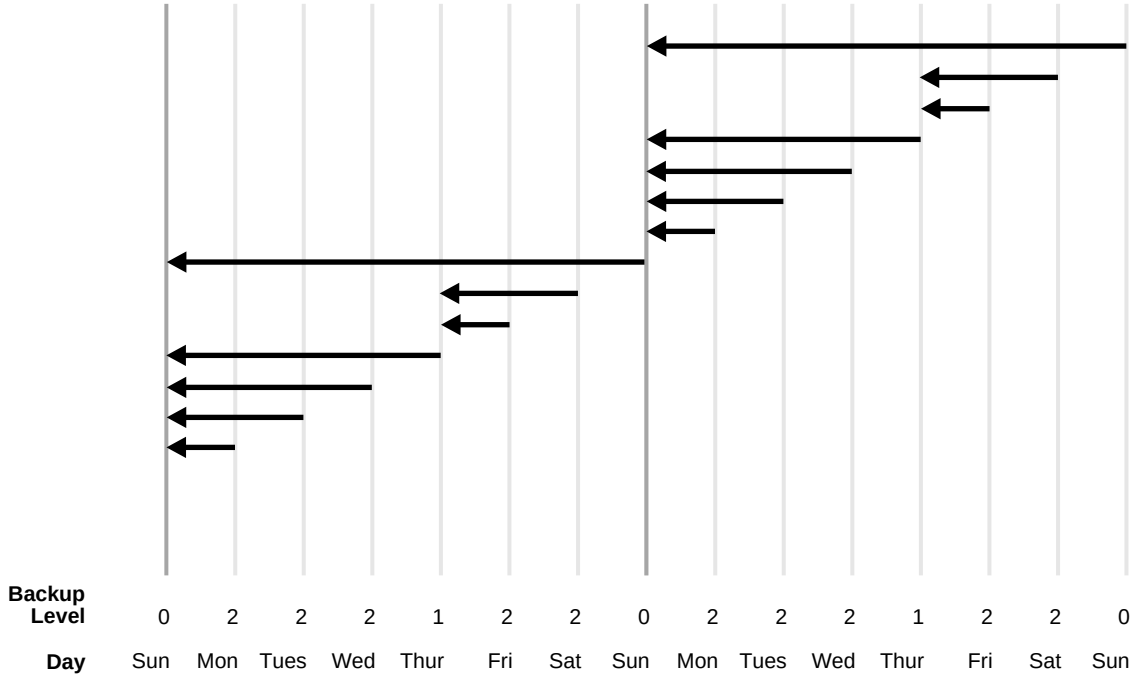
  An incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n or less; in this case the *most recent* incremental backup at level 2 or less is the level 1 Thursday backup, so only the blocks changed since the Thursday level 1 backup will be backed up.

- Saturday

  An incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n or less; in this case the *most recent* incremental backup at level 2 or less is the level 2 Thursday backup, so only the blocks changed since the Friday level 2 backup will be backed up.

- The cycle is repeated.

Figure 7–5 Cumulative Incremental Backups



In the example above:

- Sunday

  An incremental level 0 backup occurs. This backs up *all* blocks that have ever been in use in this database.

- Monday

  A cumulative incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n-1 or less; in this case the most recent incremental backup at level 2-1 or less is the level 0 Sunday backup, so only the blocks changed since Sunday will be backed up.

- Tuesday

  A cumulative incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n-1 or less; in this case the most recent incremental backup at level 2-1 or less is the level 0 Sunday backup, so all the blocks changed since Sunday will be backed up. (This includes those which were copied on Monday, as this backup is “cumula-

tive" and includes the blocks copied at backups taken at the same incremental level as the current backup).

- Wednesday

  A cumulative incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n-1 or less; in this case the most recent incremental backup at level 2-1 or less is the level 0 Sunday backup, so all the blocks changed since Sunday will be backed up. (This includes those which were copied on Monday & Tuesday, as this backup is cumulative and includes the blocks copied at backups taken at the same incremental level as the current backup).

- Thursday

  A cumulative incremental level 1 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n-1 or less; in this case the most recent incremental backup at level 1-1 or less is the level 0 Sunday backup, so all the blocks changed since Sunday will be backed up.

- Friday

  A cumulative incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n-1 or less; in this case the most recent incremental backup at level 2-1 or less is the level 1 Thursday backup, so all the blocks changed since Thursday will be backed up.

- Saturday

  A cumulative incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level n-1 or less; in this case the most recent incremental backup at level 2-1 or less is the level 1 Thursday backup, so all the blocks changed since Sunday will be backed up.

- The cycle is repeated.

Image Copies

An *image copy* contains a single file (datafile, archivelog, or control file) that you can use as-is to perform recovery. An image copy backup is similar to an O/S copy of a single file, except it is produced by an Oracle server process, which performs additional actions like validating the blocks in the file, and registering the copy in the control file. An image copy differs from a backup set because it is not multiplexed, nor is there any additional header or footer control information stored in the copy.

Image copy backups may be written only to disk.

If the image copy is of a datafile, then no restore operation is required; a **switch** command is provided to point the control file at the copy and update the recovery catalog to indicate that the copy has been "consumed;" this is equivalent to the SQL statement ALTER DATABASE RENAME DATAFILE. You can then perform media recovery to make the copy current.

An image copy can be restored to another location to prevent the "consumption" of the copy, but a restore is not strictly necessary because the datafile can be renamed to the image copy.

User-Created Image Copies

Image copies (also known as O/S copies) created by mechanisms other than Recovery Manager are supported. However, database administrators must catalog such O/S copies with Recovery Manager before using them in a restore or switch**.**

Support for both closed and open O/S copies is provided for users who have other mechanisms for creating image copies of datafiles. For example, some sites store their datafiles on mirrored disk volumes. This permits the creation of image copies by simply breaking the mirrors. After the mirror has been broken, Recovery Manager can be notified of the existence of a new copy, thus making it a candidate for use in a restore. The database administrator must notify Recovery Manager when the copy is no longer available for restore (using the **change ... uncatalog** command). In this example, if the mirror is resilvered (not including other copies of the broken mirror), you must use a **change ... uncatalog** to update the recovery catalog and indicate that this copy is no longer available.

Corruption Detection

As it is an Oracle server process which is performing a backup or copy, the server process is able to detect many types of corrupt blocks. Each new corrupt block not previously encountered in a backup or copy operation is recorded in the control file and in the alert log.

Recovery Manager queries this information at the completion of a backup and stores it in the recovery catalog. You can query this information from the control file using the views V$BACKUP_CORRUPTION and V$COPY_CORRUPTION.

If you encounter a datafile block during a backup that has already been identified as corrupt by the database, then the corrupt block is copied into the backup and the corruption is reported to the control file as either a logical or media corruption.

If a datafile block is encountered with a corrupt header that has not already been identified as corrupt by the database, then the block is written to the backup with a reformatted header indicating the block is media corrupt.

Note: Not all types of corruptions can be detected at this time.

Channel Control

A *channel* must be allocated before issuing a **backup**, **copy**, **restore**, or **recover** command. Each **allocate channel** command establishes a connection from Recovery Manager to a target database instance by starting a server process on the target instance. The **allocate channel** command also specifies the type of I/O device that the server process will use to perform the backup or restore operation. Each channel usually corresponds to one output device, unless your MML (Media Management Library) is capable of hardware multiplexing. Oracle does not recommend hardware multiplexing of Oracle backups. Note that there is only one Recovery Manager process communicating with one or many server processes attached to the target database.

You can allocate multiple channels, thus allowing multiple backup sets or file copies to be read or written in parallel by a single Recovery Manager command. Thus, the degree of parallelism within a command is affected by the number of channels (connections) that are allocated.

Each **allocate channel** command uses a separate connection to the target database. You can specify a different connect string for each channel, to connect to different instances of the target database. This is useful in an Oracle Parallel Server configuration for distributing the workload across different nodes.

Channel control commands can be used to:

- control the O/S resources Recovery Manager uses when performing **backup**, **copy, restore**, and **recover** commands
- affect the degree of parallelism (in conjunction with **filesperset**)
- specify limits on I/O bandwidth consumption (**set limit read rate**)
- specify limits on the size of backup pieces (**set limit kbytes**)
- specify limits on the number of concurrently open files (**set limit maxopenfiles**)

On some platforms, these commands specify the name or type of I/O device to use. On other platforms, these commands specify which O/S access method or I/O driver to use. Not all platforms support the selection of I/O device through this

interface; on some platforms, I/O device selection is controlled through platform-specific mechanisms.

Whether or not the **allocate channel** command actually causes O/S resources to be allocated is O/S dependent. Some operating systems allocate resources at the time the command is issued; others do not allocate resources until a file is opened for reading or writing. Furthermore, when **type disk** is specified, no O/S resources are allocated, other than for the creation of the server process.

A channel must be allocated before issuing a **change ... delete** command, which calls the O/S to delete a file. A channel which is allocated in this way is useful only for deleting files. It cannot be used as an input or output channel for a job. Only one such channel can be allocated at a time.

Parallelization

Recovery Manager may parallelize its operations, establishing multiple logon sessions and conducting multiple backup sets or file copies in parallel. Concurrent backup sets or image copies must operate on disjoint sets of database files.

Factors Affecting Degree of Parallelization

Parallelization of **backup**, **copy**, and **restore** commands is handled internally by the Recovery Manager. You only need to specify:

- more than one **allocate channel** commands
- the objects to be backed up, copied, or restored

Parallelizing Backup Set Creation

You typically must specify the number of files per set for backup sets. For example, if you allocate 20 channels, but perform a simple backup **(database)** of a database containing 20 files without any other options specified, only one channel will be used, even though 20 are allocated. If you add '**filesperset** 4', then Recovery Manager will use 5 channels (limiting each channel to 4 files). Specifying '**filesperset** 1' indicates that each channel can operate on only 1 file; thus all 20 channels will be used.

In the example above, if you back up the same database (20 files) and each file corresponds one-to-one with a tablespace, and you write the backup command as follows, you do not need to use **filesperset** to parallelize:

```
backup (tablespace 1 tablespace 2 ... tablespace 20)
```

Recovery Manager executes commands serially, that is, it completes the current command before starting the next command. Parallelism is exploited only within the context of a single command. Thus, if you want 10 datafile copies, it is better to issue a single **copy** command specifying all 10 copies rather than 10 separate **copy** commands

Parallelizing File Copies

For file copies, specify multiple datafiles to copy in a single **copy** command. For example, the following Recovery Manager script *serializes* the file copies. Only one channel is active at any one time.

```
run {
    allocate channel  c1 type disk;
    allocate channel  c2 type disk;
    allocate channel  c3 type disk;
    allocate channel  c4 type disk;
    allocate channel  c5 type disk;
    allocate channel  c6 type disk;
    allocate channel  c7 type disk;
    allocate channel  c8 type disk;
    allocate channel  c9 type disk;
    allocate channel c10 type disk;
    copy datafile 22 to '/dev/prod/backup1/prod_tab5_1.dbf';
    copy datafile 23 to '/dev/prod/backup1/prod_tab5_2.dbf';
    copy datafile 24 to '/dev/prod/backup1/prod_tab5_3.dbf';
    copy datafile 25 to '/dev/prod/backup1/prod_tab5_4.dbf';
    copy datafile 26 to '/dev/prod/backup1/prod_tab6_1.dbf';
    copy datafile 27 to '/dev/prod/backup1/prod_tab7_1.dbf';
    copy datafile 28 to '/dev/prod/backup1/prod_tab5_2.dbf';
    copy datafile 29 to '/dev/prod/backup1/prod_ndx5_1.dbf';
    copy datafile 30 to '/dev/prod/backup1/prod_ndx5_2.dbf';
    copy datafile 31 to '/dev/prod/backup1/prod_ndx5_3.dbf';
}
```

The following statement parallelizes the same example; there is one Recovery Manager command copying 10 files, and 10 channels available. All 10 channels are *concurrently active*—each channel copies one file.

```
run {
    allocate channel  c1 type disk;
    allocate channel  c2 type disk;
    allocate channel  c3 type disk;
    allocate channel  c4 type disk;
    allocate channel  c5 type disk;
    allocate channel  c6 type disk;
    allocate channel  c7 type disk;
    allocate channel  c8 type disk;
```

```
  allocate channel  c9 type disk;
  allocate channel c10 type disk;
  copy datafile 22 to '/dev/prod/backup1/prod_tab5_1.dbf',
       datafile 23 to '/dev/prod/backup1/prod_tab5_2.dbf',
       datafile 24 to '/dev/prod/backup1/prod_tab5_3.dbf',
       datafile 25 to '/dev/prod/backup1/prod_tab5_4.dbf',
       datafile 26 to '/dev/prod/backup1/prod_tab6_1.dbf',
       datafile 27 to '/dev/prod/backup1/prod_tab7_1.dbf',
       datafile 28 to '/dev/prod/backup1/prod_tab5_2.dbf',
       datafile 29 to '/dev/prod/backup1/prod_ndx5_1.dbf',
       datafile 30 to '/dev/prod/backup1/prod_ndx5_2.dbf',
       datafile 31 to '/dev/prod/backup1/prod_ndx5_3.dbf';
}
```

Multiplexed Backup Sets

A server process can concurrently operate on one or more files and/or tablespaces. If multiple files and/or tablespaces are specified, or one or more multi-file tablespaces are specified, then the datafiles are backed up concurrently and the output is multiplexed together. This means blocks from files in the same backup set will be interspersed with each other throughout the backup set.

When executing a multiplexed backup (multiple datafiles written to the same backup set), you can either partition the datafiles into backup sets explicitly, or let Recovery Manager automatically select a partitioning.

A high performance sequential output device can be kept streaming by including a sufficient number of datafiles in the backup. This is important for open database backups where the backup operation must compete with the online system for I/O bandwidth.

The control file may also be included in a datafile backup set. The Oracle logical blocksize of the objects in a multiplexed backup must be the same. Hence, datafiles and archive logs cannot be written to the same set.

Report Generation

The **report** and **list** commands provide information about backups and image copies. While the **list** command simply lists the contents of the recovery catalog, the **report** command performs a more detailed analysis of the recovery catalog. The output from these commands is written to the message log file.

The **report** command produces reports that can answer questions such as:

- what files need a backup?
- what files haven't been backed up recently?
- what backup files can be deleted?
- what files are not recoverable because of unrecoverable operations?
- what was the physical schema of the database at some previous point in time?

The **report need backup** and **report unrecoverable** commands should be used on a regular basis to ensure that the necessary backups are available to perform recovery, and to ensure that the recovery can be performed within a reasonable length of time. The **report** command lists backup sets and datafile copies that can be deleted either because they are redundant or because they could never be used by a **recover** command.

A datafile is considered *unrecoverable* if an unrecoverable operation has been performed against an object residing in the datafile subsequent to the last backup.

> **Note:** A datafile that does not have a backup is not considered unrecoverable. Such datafiles can be recovered through the use of the **create datafile** command, provided that logs starting from when the file was created still exist.

The **list** command queries the recovery catalog and produces a listing of its contents. You can use it to find out what backups or copies are available:

- backup sets containing a backup of a specified list of datafiles
- copies of a specified list of datafiles
- backup sets containing a backup of any datafile which is a member of a specified list of tablespaces
- copies of any datafile that is a member of a specified list of tablespaces

- backup sets containing a backup of any archivelogs with a specified name and / or within a specified range
- copies of any archivelog with a specified name and / or within a specified range
- incarnations of a specified database.

See Also: For more details on the Recovery Manager **report** command, see Appendix A, "Recovery Manager Command Syntax".

User Tags for Recovery Manager Backups

You can optionally assign a user-specified character string called a *tag to* backup sets and image copies (either Oracle-created copies or user-created copies). A tag is essentially a symbolic name for a backup set or file copy, and can be specified when executing the **restore** or **change** command. A tag provides a symbolic way to refer to a collection of file copies or backup sets. The maximum length of a tag is 30 characters.

Tags need not be unique: multiple backup sets or image copies can have the same tag. When a tag is not unique, then with respect to a given datafile, the tag is understood to refer to the most current suitable file (that is, the most current suitable backup containing that file may not be the most recent backup; this situation would occur when performing a point-in-time recovery). For example, if datafile copies are created each Monday evening and are always tagged "mondayPMcopy", then the tag is considered to refer to the most recently created copy. Thus, the following command switches datafile three to the most recently created Monday evening copy:

```
switch datafile 3 to datafilecopy tag mondayPMcopy;
```

Tags can indicate the intended purpose or usage of different classes of backups or file copies. For example, datafile copies that are suitable for use in a **switch** can be tagged differently from file copies that should be used only for **restore**.

Note: By default, Recovery Manager selects the most recent backups to restore, unless qualified by a tag or an **until** clause.

WARNING: You can use a tag to specify the input files to a restore or switch command. If a tag is specified, Recovery Manager will consider *only* backup sets with a matching tag when choosing which particular backup set or image copy to use.

See Also: For information about the **switch** and **restore** commands, see Appendix A, "Recovery Manager Command Syntax."

Backup Constraints

Backup operations may be performed only by an instance that has the database mounted or open. In an Oracle parallel server environment, if the instance where the backup operation is being performed does not have the database open, then the database must not be open by any instance.

Backups are supported at both the tablespace and datafile level. Backups of the control file may also be created. Backups of online logs are *not* supported. (If backups of the online logs are desired, you must run the database in ARCHIVELOG mode.)

- If the database is in NOARCHIVELOG mode, you can make whole database backups only if the database is shutdown cleanly. You can make tablespace or datafile backups if the datafiles to be backed up are in an offline normal tablespace, or tablespaces are in read-only mode.
- If the database is in ARCHIVELOG mode, then backups may be taken when the database is open and in use. Only current datafiles may be backed up. Datafiles which have been restored from backups may not be backed up until they have been made current by performing media recovery. Backups of a database that was not shutdown cleanly are allowed, but not recommended.

Recovery Manager does not back up:

- Parameter files associated with the database
- Password files associated with the database
- Operating system files
- Online redo logs

  See Also: "Backing Up Online Redo Logs" on page 3-9.

Restore Constraints

Restore operations must be performed on a started instance, however, the instance need not have the database mounted. This allows restore operations to be performed when the control file has been lost. A tablespace / datafile that is to be restored must be offline, or the database must be closed.

A restore may either overwrite the existing datafiles or its output may be directed to a new file via the **set newname** command.

Integrity Checking

Oracle prohibits any attempts to perform operations which would result in unusable backup files or corrupt restored datafiles. Among the integrity checks that the Oracle Server performs are:

- ensures that restore operations do not corrupt the database by applying backups from a previous incarnation of the database
- ensures that incremental backups are applied in the correct order
- prohibits accessing datafiles that are in the process of being restored or recovered
- allows only one restore operation per datafile at a time
- prohibits backups of unrecovered backup files
- control information stored in backups ensure that corrupt backup files are detected

Fractured Block Detection During Open Database Backups in Recovery Manager

When performing open backups *without* using Recovery Manager, tablespaces must be put in *hot backup* mode in case the operating system reads a block for backup that is currently being written by DBW0, and is thus inconsistent. This phenomenon is known as “split” or “fractured blocks.”

When performing a backup using Recovery Manager, an Oracle server process reads the datafiles, not an operating system utility. The Oracle server process reads whole Oracle blocks, and checks to see whether the block is fractured by comparing control information stored in the header and footer of each block. If a fractured block is detected, the Oracle server process rereads the block. This is why, when using Recovery Manager to back up or copy database files, tablespaces do not need to be in hot backup mode.

See Also: For information about hot backup mode, see Chapter 11, "Performing Operating System Backups".

Tracking Archive Logs

The control file tracks the location of logs archived by the ARCH background process. Each time a log is successfully archived, a record is added to the control file. The record contains the archived log filename and other relevant information.

You can specify the number of archive log records maintained when creating the control file. Any number may be specified provided that the limit on total control file size is not exceeded. These records are reused in a circular fashion. Recovery Manager queries these records and propagates them into the recovery catalog.

If there is a media failure in an online log, it may not be possible to archive it. You may therefore choose to issue an ALTER DATABASE CLEAR UNARCHIVED LOG statement. In this case, an archive log record is still created, but it indicates that no archive log exists for this log sequence number/SCN range.

Cataloging Image Copies and Archive Logs

Recovery Manager can catalog an image copy and read the meta-data. This is important when the recovery catalog is lost and you must perform disaster recovery. Only image copies and archived logs can be cataloged.

8

Performing Backup and Recovery with Recovery Manager

This chapter describes how to use Recovery Manager to manage backups, restores and recovery of your database, and includes the following topics:

- Installing the Recovery Catalog
- Maintaining the Recovery Catalog
- Using Channel Control Commands
- Generating Reports
- Maintaining Scripts
- Configuring the Snapshot Control File Location
- Backing Up Files
- Copying Files
- Restoring Files
- Switching Datafiles
- Recovering Datafiles
- Monitoring Backups and Restores

Installing the Recovery Catalog

Recovery Manager can operate with or without a recovery catalog. If you wish to use a recovery catalog, you must install it before using Recovery Manager.

To Install the Recovery Catalog

1. Create or identify the user who will own the catalog. If you wish, you can create a separate tablespace that will contain the recovery catalog.

```
create user rman identified by rman
    temporary tablespace temp
    default tablespace rcvcat
    quota unlimited on rcvcat;
 grant recovery_catalog_owner to rman;
```

2. Connect using the userid created in Step 1, and create the recovery catalog schema:

```
connect rman/rman
@?/rdbms/admin/catrman
```

3. To connect to target databases over the network (for example, non-local target databases) you will need to configure a password file. Local databases can use operating system authentication. See the *Oracle8 Administrator's Guide* for information about creating and maintaining password files.
4. Start rman

You can now use Recovery Manager.

Registering a Database

If you are creating a new database or migrating an existing Oracle7 database, issue the following command:

```
register database;
```

If Recovery Manager is installed for use with an existing version 8.x database, issue the following command:

```
register database;
```

If there are any existing user-created backups on disk that were created under version 8.x, these should be added to the recovery catalog by issuing the following command:

```
catalog datafilecopy 'filename';
```

For an Oracle7 backup to be usable for recovery in an Oracle8 database, it must have been part of a tablespace that was offline normal or read-only when the database was migrated.

Any existing archivelogs should be cataloged as follows:

```
catalog archivelog 'filename';
```

See Also: For more information about the recovery catalog, see Chapter 3, "When to Perform Backups".

Maintaining the Recovery Catalog

This section describes how to maintain and manipulate the recovery catalog, and includes the following topics:

- Registering a Target Database with the Recovery Catalog
- Resetting the Information in the Recovery Catalog
- Resynchronizing the Recovery Catalog with a Target Database
- Changing the Availability of a Backup Set or File Copy
- Deleting and Modifying from the Recovery Catalog and Operating System Backup

Registering a Target Database with the Recovery Catalog

Before using Recovery Manager with a particular target database for the first time, you must register the target database in the recovery catalog. Recovery Manager obtains all information it needs to register the target database from the target database itself. The target database must be mounted for this operation.

Oracle uses an internal, uniquely generated number called the "db identifier" to distinguish one database from another. This number is generated at the time you create the database. Typically, each database has a unique identifier; however, an exception occurs with databases created by copying files from an existing database (instead of using a **create database** statement). In these cases, the duplicate database identifiers are detected and the **register database** command fails. If this occurs, you can create a second recovery catalog in another user's schema by re-executing the **catrman.sql** script from a different Oracle userid. Then, the database

with a duplicate database identifier can be registered into the newly created recovery catalog in the new schema.

> **Note:** If you are using Recovery Manager with different target databases that have the same database name and identifier, be extremely careful to always specify the correct recovery catalog schema when invoking Recovery Manager.

Resetting the Information in the Recovery Catalog

Before you can use Recovery Manager again with a target database that has been opened with the RESETLOGS option, you must notify Recovery Manager that the database incarnation has been reset. The **reset database** command tells Recovery Manager to create a new database incarnation record in the recovery catalog. This new incarnation record becomes the "current" incarnation. All subsequent backups and log archiving done by the target database will be associated with the new database incarnation.

If you issue the ALTER DATABASE OPEN RESETLOGS command, but do not reset the database, then Recovery Manager will not access the recovery catalog because it cannot distinguish between a RESETLOGS command and an accidental restoration of an old control file. Resetting the database tells Recovery Manager that the database has been opened with the RESETLOGS option.

In the rare situation when you wish to undo the effects of opening with the RESETLOGS option by restoring backups of some prior incarnation of the database, you can use the **reset database to incarnation** *key* command to change the current incarnation to an older incarnation. You must specify the primary key of the dbinc record for the desired database incarnation. You can obtain the key value by issuing the **list incarnation of database** *identifier* command. Then, after issuing the **reset database to incarnation** command, issue the **restore** and **recover** commands to restore and recover the database files from the prior incarnation.

Resynchronizing the Recovery Catalog with a Target Database

You must resynchronize the recovery catalog on a regular basis because the recovery catalog is *not* updated automatically when a log switch occurs or when a log is archived. Instead, information about log switches and archivelogs is stored in the control file, which must be propagated periodically into the recovery catalog. How frequently you resynchronize the recovery catalog depends upon the rate at which logs are archived. The cost of the operation is proportional to the number of records in the control file that have been inserted or changed since the previous

resynchronzie. If no records have been inserted or changed, then the cost of resynchronization is very low. Thus, it is feasible to perform this operation frequently (for example, hourly) without incurring undue costs.

Note: The **backup**, **copy**, **restore** and **switch** commands update the recovery catalog automatically if:

- the target database control file is available
- the recovery catalog database is available when the command is executed

---

If the recovery catalog is unavailable during a backup or copy, you should resynchronize it.

You must also resynchronize the recovery catalog after making any change to the physical structure of the target database. As with log archive operations, the recovery catalog is *not* updated automatically when a physical schema change is made. A physical schema change includes any of the following operations:

- Adding or dropping a tablespace
- Adding a new datafile to an existing tablespace
- Adding or dropping a rollback segment

When resynchronizing the recovery catalog, Recovery Manager compares the recovery catalog to either the current control file of the target database, or a backup control file, and updates the recovery catalog with information that is missing or changed. When resynchronizing from the current control file, Recovery Manager automatically detects the records in the control file that have been updated and resynchronizes only those records. If the target database is open, then information about rollback segments is also updated in the recovery catalog (this information is used for Tablespace Point-In-Time Recovery). When resynchronizing from a backup control file, Recovery Manager *does not* verify that the backup pieces or file copies actually exist. Thus, it may be necessary to use the **change...uncatalog** command to remove records for files that no longer exist.

The following classes of recovery catalog records are resynchronized:

log history

These records are created when a log switch occurs. Recovery Manager tracks this information so that it knows what archive logs it should expect to find.

archivelog
These are records associated with archived logs that were created by archiving an online log, by copying an existing archivelog, or restoring an archivelog backup set.

backup history
These are records associated with backup sets, backup pieces, backup set members, and file copies. The **resync catalog** command updates these records when a **backup** or **copy** command is executed.

physical schema
These are records associated with datafiles and tablespaces. If the target database is open, then rollback segment information is also updated.

Physical schema information in the recovery catalog is updated only when the target database has the current control file mounted. If the target database has mounted a backup control file, a freshly created control file, or a control file that is less current than a control file that was seen previously, then physical schema information in the recovery catalog is not updated. Physical schema information is not updated when you use the **resync catalog from backup controlfile** command.

If the target database is open when you resynchronize, a new consistency point is created in the recovery catalog.

Changing the Availability of a Backup Set or File Copy

The Recovery Manager **change** command enables you to make the following record changes:

- remove references to a backup piece, datafile copy, or archivelog from the recovery catalog (**change...uncatalog**)
  - Works only when you have a recovery catalog
- remove references to a backup piece, datafile copy, or archivelog from the controlfile, recovery catalog, and physically delete the file from the operating system (**change...delete**)
  - Works with or without a recovery catalog
- remove references to a backup piece, datafile copy, or archivelog from the control file and recovery catalog when that file no longer exists on disk (**change...validate**)
  - Works with or without a recovery catalog

- mark a backup piece, datafile copy, or archivelog as unavailable (**change...unavailable**)
  - Works only when you have a recovery catalog
- mark a backup piece, datafile copy, or archivelog as available (**change...available**)
  - Works only when you have a recovery catalog

You can use the **change...uncatalog** operand to remove records that refer to a file that is no longer in the recovery catalog. It is important that you use this command to notify Recovery Manager when a backup piece, file copy, or archivelog is deleted by some means other than a **change...delete** command. **change...uncatalog** can only be used when operating with a recovery catalog.

The **delete** operand functions like the **uncatalog** operand, but in addition, the operating system or Media Manager is called to delete the backup piece or backup set. This command must be preceded by an **allocate channel for delete** command, which specifies the device type appropriate for the file being deleted.

When a large number of files have been deleted from disk, the most efficient method of reconciling the recovery catalog with what remains on disk is to use a **change...validate** command. This command validates that the specified backup pieces, file copies and archived logs that should be available on disk, are physically there; if a file is not there, the reference to that file is removed from the recovery catalog and the control file.

The **unavailable** operand is provided for cases when the file cannot be found or has migrated off site. A file that is marked unavailable will not be used in a **restore** or **recover** command. If the file is later found or returns to the main site, then it can be marked available again by using the **available** operand.

The **change** command operates only on files that are recorded in the recovery catalog (or the control file) and belong to the current database incarnation. The same is true for other commands except **catalog** and **resync from controlfilecopy**.

Deleting a Catalog Record: Example

If a cataloged file is deleted through some means other than Recovery Manager, or is otherwise lost or damaged, then you should delete the catalog record.

If the file is on disk issue the following statements:

```
change datafilecopy <primary_key> delete;
change archivelog <primary key> delete;
```

If the file is stored by a Media Manager issue the following statements:

```
allocate channel for delete type 'tape';
change backuppiece <primary_key> delete;
release channel;
```

You can obtain the primary keys of the records to be deleted by issuing a **list** command.

Deleting Many Catalog Records: Example

If you need to reconcile the recovery catalog with what is on disk when a large number of files have already been deleted, you may wish to use the following **change...validate** statements:

```
change datafilecopy <primary_key> validate;
change archivelog all validate;
```

Cataloging User-Created Backup Files

Often it is useful to make Recovery Manager aware of the existence of file copies that are created via means other than Recovery Manager.

You can use the **catalog** command to add information about a datafile copy, archivelog, or controlfile copy to the recovery catalog and control file.

You can also use the **catalog** command to catalog a datafile copy as a level 0 backup, thus enabling you to perform an incremental backup later, using the datafile copy as the base of the incremental.

Oracle 8.x continues to support the ALTER TABLESPACE BEGIN / END BACKUP command, which allows open database operating system backups. Although Recovery Manager does not create such backups, they can be added to the recovery catalog so that Recovery Manager is aware of them.

Any such backup must be accessible on disk, and must be a complete image copy of a single file. A datafile backup may be either a consistent or inconsistent whole database, tablespace or datafile backup. Recovery Manager treats all such backups as datafile copies.

For example, if datafiles are stored on mirrored disk drives, then it is possible to create an operating system copy by simply breaking the mirror. In this scenario, the **catalog** command is used to notify Recovery Manager of the existence of the operating system copy after breaking the mirror. Before the mirror is reformed, a **change... uncatalog** command should be issued to notify Recovery Manager that the file copy is being deleted.

The **catalog** command is restricted to cataloging only those files associated with the current incarnation of the database. Archivelogs and control file copies that were created from version 7.x or earlier cannot be cataloged.

Datafile copies that were created from version 7.x or earlier can be cataloged if the file belongs to a tablespace that was offline normal or read-only when the database was migrated to Oracle8.

Recovering a Lost or Damaged Recovery Catalog Database

If the recovery catalog database is lost or damaged, and recovery of the recovery catalog database via the normal Oracle recovery mechanisms is not possible, then you have two options for partially re-creating its contents:

1. Issue **catalog** commands to re-catalog archivelogs, backup control files, and datafile copies.
2. Use the **resync catalog from backup controlfile** command to extract information from a backup control file and rebuild the recovery catalog from it.

You can re-create information about backup sets *only* by using the **resync catalog from backup controlfile** command(because the **catalog** command does not support re-cataloging of backup pieces or backup sets). Recovery Manager does not verify that the files being re-cataloged still exist, so the resynchronization may add records for files that no longer exist. You should remove such records by issuing **change...uncatalog** commands.

Using Channel Control Commands

You can use channel control commands to perform the following tasks:

- Control the operating system resources Recovery Manager uses when executing **backup**, **copy**, **restore,** and **recover** commands
- Control the degree of parallelism
- Specify limits on I/O bandwidth consumption and the size of backup pieces

Each channel allocation establishes a connection from Recovery Manager to a target database instance. Multiple channels can be allocated simultaneously, thus allowing multiple backup sets or file copies to be read or written in parallel by a single job. Thus, the degree of parallelism within a job is controlled by the number of channels (connections) that are allocated.

Each channel allocation uses a separate connection. You can specify a different connect string for each channel. This is useful in an OPS (Oracle Parallel Server) configuration for distributing the workload across different nodes.

Whether or not the **allocate channel** command actually causes operating system resources to be allocated is operating system dependent. On some platforms, operating system resources are allocated at the time the command is issued. On other platforms, operating system resources are not allocated until a file is opened for reading or writing. Furthermore, when **type disk** is specified, no operating system resources at all are allocated by this command.

At least one **allocate channel** command must precede a **backup**, **copy**, **restore** or **recover** command. The **copy** command can use only channels that specify **type disk**. Other devices types that happen to be allocated when a **copy** command executes are ignored.

See Also: For more information about channel commands, see your operating system-specific Oracle documentation.

Channel Control Commands

allocate channel

The **allocate channel** command establishes a connection to a target database instance. Each connection operates on one backup set at a time (for **backup**, **restore**, or **recover**) or one file copy at a time (for **copy**). If multiple connections are established, then each connection operates on a separate backup set or file copy.

If a backup set is to be written to or read from disk, then you must use the **type disk** allocate operand. Otherwise, it is assumed that the device is a sequential I/O device (for example, tape). Copies are always written to disk, therefore they can use only channels that have **type disk** specified.

The allocated channel must be given a name (*channel_id*) that is used when releasing the channel and when reporting I/O errors. The *channel_id* is any blank-delimited character string, other than a keyword, that is meaningful to the user.

Following are descriptions of **allocate channel** operands:

type

This operand specifies either **disk** or a quoted string. If **disk** is specified, then all backup sets written or read by this connection are on disk. If a quoted string is specified, then it is assumed to be a platform-specific specification of some type of sequential I/O device or access method.

The exact syntax and semantics of sequential I/O device types are platform-specific. If you do not specify this operand, then you must specify the **name** operand to identify a particular sequential I/O device.

name

This port-specific string specifies the name of the sequential I/O device to allocate. If you do not specify, then any available device of the specified type is used. Do not use this operand if **type disk** is specified.

parms

This port-specific string specifies parameters regarding the device to allocate. Do not use this operand if you've specified **type disk**.

connect

This parameter specifies a connect string to the target database instance where Recovery Manager should conduct the backup or restore conversation. This parameter is intended for use by OPS installations that want to spread the work of backup or restore across different OPS instances. If you do not specify this parameter, then all conversations are conducted on the target database instance specified by the **target** command-line parameter.

format

This parameter specifies the format to use for the names of backup pieces that are created on this channel. This format will only be used if no **format** option is specified in the **backup** command. See the description of the **format** operand of the **backup** command.

release channel

The **release channel** command releases a sequential I/O device. However, the connection is maintained for additional channel allocation commands. The operand is simply the *channel_id* specified when the channel was allocated.

set limit channel

The **set limit channel** commands specify limits that apply to any **backup** or **copy** command that executes using this device.

Following are descriptions of s**et limit channel** operands:

readrate

This limit specifies the maximum number of buffers (each of size db_blocksize * db_file_direct_io_count) per second, which will be read for backup or copy from each of the input datafiles. Use this parameter to ensure that the command does not consume excessive disk bandwidth, and thereby degrade online performance.

kbytes

This specifies the maximum size of the backup pieces created on this channel.

maxopenfiles

This limit controls the maximum number of input files that a backup will have open at any particular instant. This parameter is used for preventing "too many open files" operating system errors when backing up a large number of files into a single backup set.

If **maxopenfiles** is not specified, then a maximum of 32 input files may be open concurrently.

Generating Reports

You can use the **report** command to produce two kinds of reports that answer questions such as the following:

- What files need a backup?
- What files haven't had a backup in a while?
- What files are not recoverable due to unrecoverable operations?
- What backup files can be deleted?
- What was the physical schema of the database at some previous point in time?

The **list** command queries the recovery catalog and produces a listing of its contents. The following information may be listed:

- backup sets containing a backup of a specified list of datafiles
- copies of a specified list of datafiles
- backup sets containing a backup of any datafile which is a member of a specified list of tablespaces

- copies of any datafile that is a member of a specified list of tablespaces
- all backup sets or copies of all datafiles in the database, optionally restricted by time, datafile copy filename, device name, or tag
- backup sets containing a backup of any archivelogs with a specified name and / or within a specified range
- copies of any archivelog with a specified name and / or within a specified range
- incarnations of a specified database or of all databases known to the recovery catalog

The **report** and **list** commands produce similar output. The **list** commands simply list the contents of the recovery catalog. The **report** commands perform more detailed analyses of the recovery catalog. The output from the **report** and **list** commands is written to the message log file.

The **report need backup** and **report unrecoverable** commands should be used on a regular basis to ensure that the necessary backups are available to perform recovery, and to ensure that the recovery can be performed within a reasonable length of time. The **list** commands can be used to query what backups or copies are available.

Generating a Report

You can use the **report** command to produce one of the following types of reports:

- **report unrecoverable** *report_object_list device_type_list*
- **report need backup** *need_backup_operand report_object_list device_type_list*
- **report obsolete** *report_obsolete_operand_list device_type_list*
- **report schema** *at_clause*

Where:

report_object_list

Specifies the datafiles to be reported on. The report can include the entire database (optionally skipping certain tablespaces), a list of tablespaces, or a list of datafiles.

device_type_list

Can be used to limit the backup sets that will be considered when deciding whether or not a file is unrecoverable or is in need of a more recent backup. If specified, only backup sets residing on one of the

specified device types will be considered. If not specified, all available backup sets will be considered. Datafile copies will always be considered.

Reporting Unrecoverable Datafiles

You can use the **report unrecoverable** command to list all datafiles that are unrecoverable. A datafile is considered unrecoverable if an UNRECOVERABLE operation has been performed against an object residing in the datafile since the last backup of the datafile.

Note that the non-existence of any backup of a datafile is not sufficient reason to consider it unrecoverable. Such datafiles can be recovered through the use of the **create datafile** command, provided that logs starting from when the file was created are still in existence.

Example: Datafiles Requiring a New Backup The following command lists all datafiles that cannot be completely recovered from the existing backups because redo may be missing:

```
report unrecoverable;
```

Reporting Datafiles that Need to be Backed Up

The **report need backup** command lists all datafiles that need a new backup. The report assumes that the most recent backup would be used in the event of a restore.

Following are descriptions of *need_backup* operands:

incremental

An integer specifying a threshold number of incremental backups. If complete recovery of a datafile would require the application of more than this many incremental backups, then the datafile is considered in need of a new full backup. This assumes the most efficient strategy, which is to use the lowest level of incremental backup whenever there is a choice. This is the same strategy that would be used if the file were actually being recovered by the **recover** command. Note that files for which no backups exist will not appear in this list. They can be found by using the **report need backup days** command.

days

An integer specifying a threshold number of days of log files that will need application during recovery of this file. For online files, this is the number of days since the last full or incremental backup of a file. The time of day is not considered when calculating the age of a backup set (i.e. a backup taken anytime yesterday is 1 day old). If multiple copies of a backup set exist, the completion time of the original backup set is used. If the most recent backup of this file is older than this number of days, then the file is in need of a new backup.

If the target database controlfile is mounted and current, the following optimizations will be made to this report:

1. Files that are offline and whose most recent backup contains all changes to the file will not be included.

2. Files that were offline and are now online, and whose most recent backup contains all changes up to the offline time, will only be reported if they have been online for more than the specified number of days.

redundancy

An integer specifying the minimum level of redundancy considered necessary. **redundancy 2** means that there must be at least 2 backups of each datafile for it for it to be considered not in need of a backup.

Example: Datafiles That Need to be Backed Up The following command reports all datafiles in the database that would require the application of three or more incremental backups to be recovered to their current state:

```
report need backup incremental 3 database;
```

The following command reports all datafiles from tablespace “system” that haven’t had a backup (full or incremental) in 5 or more days:

```
report need backup days 5 tablespace system;
```

Reporting Obsolete Datafile Backups

You can use the **report obsolete** command to list backup sets and datafile copies that can be deleted because they are redundant.

A backup is obsolete if it meets one of the following criteria:

- It is a backup of a file which no longer exists, or did not exist at the specified *until-time*
- n more recent backups of the same file exist and are available, where n is the desired redundancy
- It is from an orphan incarnation, if the **orphan** option is specified

Following are descriptions of *report_obsolete* operands:

redundancy

An integer specifying the minimum level of redundancy considered necessary. If more than this many full backups or copies exist for a given datafile, then the remainder of the backups are obsolete. This must be non-zero. The default value is one.

orphan

Specifies that backups and copies that can never be used (because they belong to incarnations of the database that are not predecessors of the current incarnation), will be considered obsolete.

until-clause

If an until-clause is specified, then no backup will be considered obsolete or redundant if it contains any changes beyond the specified time. This is useful if the database must be recoverable to a point in time earlier than the present time.

Reporting Datafiles at the Current Time

The **report schema** command lists the names of all datafiles and tablespaces at the specified point in time, or at the current time. A point in time may be specified as a time, an SCN, or a redo log.

The ***at_clause*** has the following structure:

at time

A quoted string specifying the time.

at scn

An integer representing the SCN is specified.

at logseq *integer* thread *integer*

Integers that specify the log sequence number and thread number to consider. This is the time when the specified log and thread were first opened.

Generating Lists

You can use the **list** command to produce a detailed report of all information about a specified group of backup sets or copies known to the recovery catalog. The following list commands are available:

- **list copy of** *list_object_list list_qualifier_list*

  List information about datafiles copies and archivelogs.

- **list backupset of** *list_object_list list_qualifier_list*

  List information about backup sets.

- **list incarnation of** *identifier list_qualifier_list*

  List information about the incarnations of a database. The listing will include the primary keys of all database incarnation records for the specified database name. The key can then be specified in a **reset database** command to change the incarnation that Recovery Manager considers to be current to a previous incarnation. If no identifier is specified, then all databases registered in the recovery catalog are listed.

list_object_list

This specifies the tablespaces, datafiles, or archivelogs whose backup sets or copies are to be listed.

database *skip_clause*

Backup sets or datafile copies of all files in the current database are listed. Optionally, tablespaces may be skipped by using the *skip_clause*.

tablespace

A list of tablespace names. Backup sets or datafile copies that include at least one datafile from a specified tablespace are listed.

datafile

A list of datafile names or numbers. Backup sets or datafile copies that contain at least one of the specified datafiles are listed.

archivelog_record_specifier

list_qualifier_list

List qualifiers are specifications that can be used to limit the objects whose backup sets or copies are to be listed:

tag ***tag***

Datafile copies and backup sets may be restricted by specifying the tag of the copy or backup. If *tag* is specified, only copies/backups with the specified tag will be listed.

like ***file_name_pattern***

Datafile copies and archived logs may be restricted by specifying a file name pattern. The pattern may contain Oracle pattern matching characters '%' and '_'. If *file_name_pattern* is specified, only files whose name matches the pattern will be listed.

from/until ***time***

All files may be qualified with a time range. If **from** or **until** is specified, only copies or backups that completed within the specified time period will be listed. Either or both of these options may be specified.

device_type_list

This option applies only to the **list backupset** command. If specified, only backup sets residing on one of the specified device types will be listed. If not specified, all available backup sets will be listed.

Example: Listing Backups of a Datafile

The following command lists all known backups of datafile '?/dbs/foo.f':

```
list backupset of datafile "?/dbs/foo.f";
```

Example: Listing Copies of the Datafiles of a Tablespace

The following command lists all copies of datafiles in tablespace "system":

```
list copy of tablespace system;
```

Maintaining Scripts

A stored script is a named entity that you can assign any meaningful name. The **execute script** command is used to execute a stored script. The **execute script** command is legal only within a *job_command_list*.

Four stored script commands are available:

- create a stored script
- replace a stored script
- delete a stored script
- print a stored script to the log file (CLI) or a screen

The stored script feature is provided primarily to provide a common repository for frequently executed collections of Recovery Manager commands. For example, the Recovery Manager commands needed to perform nightly backups can be collected into a single script called “nightlybackup”. Storing the script in the recovery catalog instead of in an operating system text file has the advantage that it is accessible to any database administrator using Recovery Manager, regardless of which machine Recovery Manager is executed upon.

Creating and Replacing Scripts

The **create** or **replace** commands either create or replace a stored script and store it in the recovery catalog for future reference. The script is not executed immediately. The **execute script** command must be used to execute the stored script. The **replace script** command also creates a script if one doesn’t already exist.

Deleting Scripts

Use the **delete script** command to delete a stored script from the recovery catalog.

Printing Scripts

Use the **print script** command to print a stored script to the Recovery Manager log file.

Configuring the Snapshot Control File Location

When Recovery Manager needs to read a read-consistent version of the control file, it creates a temporary backup of the control file. By default, the location the name and location of the snapshot control file is port specific.

However, it is possible to choose the name and location this file is written to by using the **set snapshot controlfile name to** command. Any subsequent snapshot control files will be created according to the name and location specified in the command. For example:

```
set snapshot controlfile name to '/oracle/dba/prod/snap_prod.ctl';
```

It is also possible to set the snapshot control file name to a raw device. This is important for OPS databases where more than one instance in the cluster will use Recovery Manager; this is because server processes on each node must be able to create a snapshot control file with the same name and location. For example:

```
set snapshot controlfile name to '/dev/vgd_1_0/rlvt5';
```

See Also: *Oracle8 Parallel Server Concepts and Administration*.

Backing Up Files

When backing up files, the target database must be started and mounted. The control file must be current—not a backup control file.

You must give each backup piece a unique name using the **format** operand. Several substitution variables are available to aid in generating unique names. You can specify the **format** operand in the **backup** command, in the *backup_specification* level, or in the **allocate channel** command.

You can also limit the number of files to place into a single backup set. Generally, for datafile or datafile copy backups, you should group multiple datafiles into a single backup set to the extent necessary to keep an output tape device streaming, or to prevent the backup from consuming too much bandwidth from a particular datafile. The fewer files there are in a backup set, the faster one of them can be restored, since there is less data belonging to other datafiles that must be skipped. For archivelog backup sets, it is advisable to group logs from the same time period into a backup set because it is likely that they will need to be restored at the same time.

I/O errors encountered when reading files or writing to the backup pieces cause jobs to be aborted. The backup pieces that were being written at the time of the error will need to be re-written from the beginning. Any backup sets that were successfully written prior to the abort are retained.

Datafile Backup Sets

If the database is in ARCHIVELOG mode, then the target database can be open or closed. It is not necessary for the database to be closed cleanly. If the database is in NOARCHIVELOG mode, then it must be closed cleanly prior to taking a backup.

> **Note:** If the database is in ARCHIVELOG mode, it is not necessary to shutdown cleanly for a cold backup. However, Oracle recommends you do so that the backup is consistent.

You can also back up offline or read-only tablespaces.

Corrupt datafile blocks are identified by the server process as corrupt and written out to the backup. Oracle records the address of the corrupt block and the type of corruption in the control file. You can access these control file records in the V$BACKUP_CORRUPTION view. You can specify the maximum number of corruptions allowed in a datafile being backed up using **set maxcorrupt**.

set maxcorrupt

This sets a limit on the number of previously undetected block corruptions that will be allowed in a specified datafile or list of datafiles. If a backup or copy command detect more than this number of corruptions, then the command is aborted. The default limit is zero, meaning no corrupt blocks will be tolerated.

Performing Backups

Use the **backup** command to create one or more backup sets. Each resulting backup set contains one or more datafiles, datafile copies, or archivelogs from the target database. You can also place a backup of the control file into a datafile backup set. A file cannot be split across different backup sets. Archivelogs and datafiles cannot be mixed into a single backup set.

The number of backup sets produced during a backup depends on the number of *backup_specifications* in the command, the number of files specified or implied in each *backup_object_list*, and the value of the FILESPERSET limit. Each *backup_specification* produces at least one backup set. If the number of files specified or implied in its *backup_object_list* exceeds the FILESPERSET limit, then the *backup_specification* will produce multiple backup sets. If no limit is specified, then each *backup_specification* produces exactly one backup set.

If multiple backup sets are to be created and multiple channels are allocated, then Recovery Manager automatically parallelizes its operation and writes multiple

backup sets in parallel. A single backup set cannot be striped across multiple channels. Recovery Manager automatically assigns a backup set to a device. It is possible to specify that all backup sets for a *backup_specification* be written to a specific channel.

Types of Recovery Manager Backups The *backup_type* applies to all *backup_specifications* in the *backup_specification_list*. The following two backup types are available:

full

This is the default if neither **full** nor **incremental** is specified. A full backup copies all blocks into the backup set, skipping only datafile blocks that have never been used. No blocks are skipped when backing up archivelogs or control files.

A full backup has no effect on subsequent incremental backups, and is not considered to be part of the incremental backup strategy.

incremental

An incremental backup at a level greater than 0 copies only those blocks that have changed since the last incremental backup. An incremental backup at level 0 is identical in content to a full backup, but the level 0 backup is considered to be part of the incremental strategy.

Certain checks are performed when attempting to create an incremental backup at a level greater than zero. These checks ensure that the incremental backup would be usable by a subsequent **recover** command. Among the checks performed are:

- A level 0 backup set must exist, or level 0 datafile copies must exist for each datafile in the **backup** command. These must also not be marked unavailable.

- Sufficient incremental backups taken since the level 0 must exist and be available such that the incremental backup about to be created could be used.

Multiple levels of incremental backup are supported. A level N incremental backup copies only those blocks that have changed since the most recent incremental backup at level N or less.

If **incremental** is specified, then all *backup_object_lists* in the command must specify one of the following: **datafile**, **datafilecopy**, **tablespace**, or **database**. Incremental backups of control files, archivelogs or backup sets are not supported.

Backup Command Operand List

You can specify a number of operands that apply to the entire **backup** command. Some of these operands may also be specified at the *backup_specification* level.

tag

A backup set can be given a user-specified identifier called a *tag*, which is a character string that is not a reserved word, typically with a meaningful name like "monday_evening_backup" or "weekly_full_backup". Tags must be 30 characters or less.

The syntax allows specification of the tag at the *backup_command* level or the *backup_specification* level. If specified at the command level, then all backup sets created by this command are given this tag. If specified at the *backup_specification* level, then backup sets created as a result of different backup specifications can have different tags. If specified at both levels, then the tag in the *backup_specification* takes precedence.

cumulative

Causes an incremental backup to re-copy all the blocks that the previous backup at the same level copied, in addition to those blocks that have changed in the interim.

nochecksum

This suppresses block checksums. Unless this option is specified, a checksum is computed for each block and stored in the backup. The checksum is verified when restoring from the backup and also written to the datafile when restored.

If the database is already maintaining block checksums, then this flag has no effect. The checksum is always verified and stored in the backup in this case.

filesperset

This specifies the maximum number of files to place in one backup set. If the number of files specified or implied by the backup specification is greater than **filesperset**, then the backup specification will cause multiple backup sets to be created.

When you specify this parameter, Recovery Manager uses the minimum value of the calculated and specified value, which ensures that all devices are used. If you do not specify filesperset, the Recovery Manager uses the minimum value of the calculated value and 64, again

ensuring that all channels are used, but limiting the number of files in a backup set.

Recovery Manager always attempts to create enough backup sets so that all allocated channels have work to do. An exception to this occurs when there are more channels than files to back up.

setsize

This enables users to specify a maximum size for a backup set. The limit is specified in units of 1K (1024 bytes). Thus, to limit a backup set to 3Mb, you would specify **setsize**=3000. Recovery Manager attempts to limit all backup sets to this size. This is a particularly useful option when you wish to make each backup set no larger than 1 tape.

Backup Specification List

A *backup_specification_list* contains a list of one or more *backup_specification*s. A *backup_specification* minimally contains a list of objects to backup and a *format* operand to specify a filename template for the backup pieces.

Each *backup_specification* creates one or more backup sets. A *backup_specification* will cause multiple backup sets to be created if the number of datafiles specified in or implied by its *backup_object_list* exceeds the **filesperset** limit.

Backup Object List

Each *backup_specification* contains exactly one *backup_object_list*. The *backup_object_list* specifies which objects to backup.

database

Indicates all datafiles, in addition to the control file, are backed up.

tablespace

This specifies a list of one or more tablespaces to back up. All datafiles that are currently part of the tablespaces will be backed up. Any number of tablespaces can be specified.

The keywords **database** and **tablespace** are provided merely as a convenience. These forms are translated internally into a list of datafiles.

datafile

This specifies a list of one or more datafiles to back up. Datafiles can be specified either by filename or by datafile number. If a filename is specified, then it must be the name of a current datafile as listed in the recov-

ery catalog (when a recovery catalog is used); otherwise, as listed in the control file.

If file1 (the first file of the system tablespace) is backed up, the control file is automatically included.

datafilecopy

This specifies a list of one or more datafile copies to back up. The files can be specified either by filename or by tag. If specified by tag, and multiple datafile copies with this tag exist, then only the most current datafile copy of any particular datafile is backed up.

archivelog

This specifies a filename pattern, and/or a time-range or log sequence range used to choose which archivelogs to include in a backup. All archivelogs that meet the specification are included in the backup. If the range is specified by time, then the logs that were current at the begin and end times are included in the backup.

current control file

This specifies that the current control file is backed up.

backup controlfile

This specifies the filename of a backup control file to back up.

backupset

This specifies the primary key of a backup set to back up. The backup set must be on disk.

Backup Operand List

This is a list of operands specifying attributes for the backup sets and backup pieces that are to be created for this *backup_specification*.

tag

If specified, the backup sets are given the specified tag. The tag value is **null** otherwise.

parms

This is a quoted string containing OS-specific information. The string is passed to the OSD layer each time a backup piece is created.

format

This specifies the file name to use for the backup pieces. The name must be enclosed in quotation marks. Any name that is legal as a sequential filename on the platform is allowed, provided that each backup piece gets a unique name. If backing up to disk, then any legal disk filename is allowed, provided it is unique.

The **format** operand may be specified in any of these places:

- the *backup_specification*

- the **backup** command

- the **allocate channel** command

If specified in more than one of these places, Recovery Manager will search for the format operand in the order shown above.

The following substitution variables are available in format strings to aid in generating unique filenames:

%p

The backup piece number within the backup set. This value starts at 1 for each backup set and is incremented by 1 as each backup piece is created.

%s

The backup set number. This is a counter in the control file that is incremented for each backup set. The counter value starts at 1 and is unique for the lifetime of the control file. If a backup control file is restored, then duplicate values may result. Also, CREATE CONTROLFILE initializes the counter back to 1.

%d

The database name.

%n

The padded database name.

%t

The backup set stamp. This is a 4-byte value derived as the number of seconds since a fixed reference date / time. The combination of %s and %t can be used to form a unique name for the backup set.

%u

An 8-character name composed of compressed representations of the backup set number and the time the backup set was created.

include current control file

This operand creates a snapshot of the current control file and places it into each backup set produced by this *backup_specification*.

filesperset

This specifies the maximum number of datafiles to place in one backup set. If the number of datafiles specified or implied by the backup specification is greater than **filesperset**, then the backup specification creates multiple backup sets.

channel

The name of a channel to use when creating the backup sets for this *backup_specification*. If this operand is not specified, then Recovery Manager dynamically assigns the backup sets for this *backup_specification* to any available channels during job execution.

delete input

This operand causes the input files to be deleted upon successful creation of the backup set. May be specified only when backing up archived logs or datafile copies. It is equivalent to issuing a **change...delete** command for all of the input files.

Backing Up: Scenario

Let's assume there is a database called FOO that a database administrator wants to backup. The administrator has 3 tape drives available for the backup, and the database has 26 datafiles in it. The administrator wants to multiplex the backup, placing 4 files into each backup set, so chooses the number 4 because it is sufficient to keep the tape drive streaming. The administrator is not concerned about how datafiles are grouped into backup sets.

The administrator issues the following commands:

```
create script foo_full {
   allocate channel t1 type 'SBT_TAPE';
   allocate channel t2 type 'SBT_TAPE';
   allocate channel t3 type 'SBT_TAPE';
backup full filesperset 4
   database format 'FOO.FULL.%n.%s.%p');
run {
```

```
    execute script foo_full;
}
```

This script will back up the whole database, including all datafiles and the control file. Since there are 27 files to be backed up (26 datafiles and a control file) and a maximum of 4 files per backup set, 7 backup sets will be created. The backup piece filenames will have the following format: FOO.FULL.*database_name.x.y*. Assuming no backup sets have been recorded in the recovery catalog prior to this job, then *x* will range from 1 through 7, and *y* will start at 1 for each backup set and will increment as backup pieces are created.

Copying Files

In many cases, copying datafiles can be more beneficial than backing them up, because when you copy files, the output is suitable for use without any additional processing. In contrast, a backup set must be processed by a **restore** command before it is usable. So, you can perform media recovery on a datafile copy, but not directly on a backup set, even if it contains only one datafile and is composed of only a single backup piece.

Use the **copy** command when you wish to create a copy of a file. The output file is always written to disk.

The following types of files can be copied:

- a current datafile
- a datafile copy created by a previous COPY command or by some other means
- an archivelog
- the current control file
- a backup control file

Copy Command Specifiers

The **copy** command has one or more *copy_specifiers*, each of which specify one input file and one output file. At least one **allocate channel** command specifying **type disk** must precede a **copy** command.

If the **copy** command contains multiple *copy_specifiers* and multiple channels are allocated, then it is executed in parallel, with the degree of parallelism determined by the number of **allocate channel** commands.

Following are descriptions of the **copy** command specifiers:

datafile

This makes a copy of a current datafile. The datafile may be specified either by filename or filenumber. If the filename is used, then the filename must be the name of a datafile listed in the control file.

datafilecopy

This makes a copy of an existing datafile copy. The existing copy may have been created by either a previous **copy** command or by some external operating system facility. The input file can be specified by filename or tag. The filename must *not* be the name of a current datafile listed in the control file.

archivelog

This makes a copy of an archive log. The archive log may have been created by the Oracle log archiving process or by a previous **copy** command. You must specify the archive log by filename.

current control file

This makes a copy of the current control file.

backup control file

This makes a copy of an existing backup control file. You can specify the control file either by filename or by tag.

Optionally, you can supply the following keywords to the copy command:

tag

When you specify this option, the output file is assigned the specified tag.

level 0

When you specify this option, the datafile copy is included in the incremental backup strategy, and thus can serve as a basis for subsequent incremental backup sets. If you do not specify this option, the datafile copy has no impact on the incremental backup strategy.

Restoring Files

With Recovery Manager you can restore datafiles from backup sets or from copies on disk. The restore may be directed at either the current datafile location (overlaying the file currently there) or to a new location (by using the **set newname** command). If datafiles are restored to a new location, then they are considered datafile copies and are recorded as such in the control file and recovery catalog.

It is also possible to restore backup sets containing archive logs, which is not normally necessary because Recovery Manager performs this automatically as needed during recovery. However, you can improve the recovery performance by pre-restoring archive log backup sets that will be needed during the recovery.

Database Point-In-Time Recovery

An easy way of specifying the time to restore and recover to, is by using the **set until** command. This command affects any subsequent restore, switch and recover commands that are in the same **run** command:

set until

This specifies a point in time for a subsequent restore or recover command. The point in time may be specified using the following key words:

- time *string*

- logseq *integer* **thread** *integer*

- scn *integer*

If the **time** keyword is specified then *string* must be a formatted according to the NLS date format specification currently in effect. This format is specified by the NLS_DATE_FORMAT environment variable on most platforms.

File Selection

Recovery Manager uses the recovery catalog (or target database control file if no recovery catalog is available) to select the best available backup sets or copies for use in the restore. Preference is given to copies rather than backup sets, and when multiple choices are available, the most current backup sets or copies are used, taking into account the *until_clause* if specified.

All specifications (*from_tag, from_type*, and *until_clause*) must be satisfied before a backup set or file copy is selected for restoration. Restore also considers the device

types of the allocated channels when performing automatic selection. If no available backup or copy in the recovery catalog satisfies all the specified criteria, then Recovery Manager returns an error during compilation of the restore job. If the file cannot be restored because no backup sets or datafile copies reside on media compatible with the device types allocated in the job, then cancel the job. You can then create a new job specifying **channel allocation** for devices that are compatible with the existing backup sets or datafile copies.

Restore Destination for Datafiles

By default, the **restore** command restores datafiles to their current location as specified in the recovery catalog (for example, the current datafiles are overlaid). If this is not desired, then issue **set newname** prior to restoring. In this case, the restored datafiles are considered datafile copies, and you must perform a switch to make them the current datafiles. The specified filename is created or overwritten if it already exists.

set newname

This sets the default name for all subsequent **restore** or **switch** commands that affect the specified datafile. If this command is not used prior to a datafile restore, then Recovery Manager restores the file to its default location, as explained above.

Restore Destination for Control Files

You must specify a destination name when restoring a control file. The specified filename is created or overwritten if it already exists.

Replicating Control Files

You can use the **replicate** command to copy a control file to multiple destinations. You specify the input control file by name, and the output destination files in the **control_files** initialization parameter of the target database.

You can use the **replicate** command following a **restore** command, which has restored the control file, to prepare the database for mounting. This is equivalent to multiple **copy controlfile** commands. At least one **allocate channel** command specifying **type disk** must precede a **replicate** command.

Restore Destination for Archived Logs

Archived logs are restored to files whose names are constructed using the LOG_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT initialization parameters of the target database. These parameters are combined in a port-specific fashion to derive the name of the restored archived log file. You can override LOG_ARCHIVE_DEST by specifying the archive log destination prior to restoring by using the **set archivelog destination to** command. The restored archived logs are created or overwritten if they already exist.

Using **set archivelog destination**, it is possible to manually stage many archived logs to many different locations while a database restore is occurring. Recovery Manager knows where to find the newly restored archive logs; it does not require them to be in the LOG_ARCHIVE_DEST for recovery to find them.

set archivelog destination

This overrides the **log_archive_dest** initialization parameter in the target database when forming names for restored archive logs during subsequent **restore** and **recover** commands.

For example, if you specify a different destination than the one in the init.ora LOG_ARCHIVE_DEST parameter and restore archivelog backups, subsequent restore and recovery operations will detect this new location and will not look for the files in LOG_ARCHIVE_DEST.

Guidelines for Restoring Datafiles

A restore restores full datafile backup sets, incremental level 0 backups or datafile copies. Incremental backups at levels greater than 0 are *not* restored via the **restore** command. Instead, you would perform a recovery to apply an incremental backup to a level 0 backup. You typically restore when a media failure has damaged the current copy of a datafile. You also restore prior to performing a point-in-time recovery.

If you issue **set newname** commands to restore datafiles to a new location, with the intention of performing a recovery afterwards, then you must perform a switch after restoring but before recovering to make the restored datafiles the current datafiles. A *to_specifier* clause is required when restoring the control file: there is no default location to which the control file is restored.

If you do not specify **set newname** when restoring datafiles, then either the database must be closed, or the datafiles must be offline. If the entire database is to be restored, then it must be closed.

If the target database is mounted, then its control file will be updated with any applicable datafile copy and archived log records to describe the restored files.

At least one **channel allocation must** precede a restore. If you use the **from copy** operand, then the allocated channels should be of **type disk**. If using the **from backup** operand, then the appropriate type of sequential I/O devices must be allocated for the backup sets that will need restoration. If the appropriate type of device is not allocated, then you may not be able to find a candidate backup set or copy for restoration, and the **restore** fails.

Files within a single restore are restored in parallel if multiple channels are allocated, with the degree of parallelism controlled by the number of **allocated channels**. You can restore files in separate *restore_specifications* from the same backup set if the best candidates for restoration are on the same backup set and there are no conflicting PARM or CHANNEL operands on either of the respective *restore_specifications*.

Restore Command Operand List

The following operands apply to the **restore** command. Some of these operands can also be specified at the *restore_specification* level.

from backupset/datafilecopy

This specifies whether a restore should restore from file copies on disk or from backup sets. If this operand is not specified, then the **restore** chooses the most recent backup set or file copy. “Most recent” is the file copy or backup set that needs the least media recovery.

until

In the absence of any other criteria, Recovery Manager will select the most current file copy or backup set to restore. If this is not desired, then an **until** clause may be specified to limit the selection to those backup sets or file copies that would be suitable for performing a point-in-time recovery to a specified time.

tag

By default, a restore chooses the most recent backup set or file copy available. This automatic selection can be overridden by specifying a *from_tag*. The *from_tag* restricts the automatic selection to backup sets or file copies that have a specified tag. If multiple backup sets or file copies are available with a matching tag, then the most recent one is selected.

channel

The name of a channel to use for this restore. If no channel is specified, then restore will use any available channel which is allocated with the correct device type to restore the required files.

parms

This is a quoted string containing operating system specific information. The string is passed to the OSD layer each time a backup piece is restored.

Restore Specification List

A *restore_specification_list* consists of one or more *restore_specifications*. Each *restore_specification* contains a list of objects to restore and, optionally, restore options that will override the options from the *restore_command_operand_list*.

Restore Specification

Each *restore_specification* contains exactly one *restore_object_list*. The **restore** operates against a list of *restore_objects*.

database

All datafiles in the database will be restored. Note that, unlike a backup, this does not also include the control file. You can use an optional **skip** argument to skip restoring certain tablespaces. This is useful for avoiding restoration of tablespaces containing only temporary data.

tablespace

All datafiles in the specified tablespaces will be restored.

datafile

The specified datafiles will be restored.

controlfile

The control file will be restored to the specified location.

archivelog

The specified archive logs will be restored.

The tablespace and database forms of *restore_object* are provided simply for convenience. They are translated into the corresponding list of datafile numbers.

It is a mistake to specify a datafile more than once in one restore job. For example, the following is considered illegal since datafile 1 is specified both explicitly and implied by the system tablespace:

```
restore (tablespace system) (datafile 1);
```

Switching Datafiles

When you *switch* datafiles, you are converting a datafile copy into a current datafile. By "current datafile" we mean the file that the control file points to (for example, the filename of the datafile copy becomes the new filename of the datafile as listed in the control file). Media recovery will be required for the datafile.

You should switch datafiles when you want to have a datafile copy become the current version of a datafile. This is equivalent to using the ALTER DATABASE RENAME DATAFILE command. Note that this effectively causes the location of the current datafile to change. Also note that switching "consumes" the copy. The corresponding records in the recovery catalog and the control file are deleted.

The datafile that is the target of the switch can be specified either by filename or by filenumber. You can specify the datafile copy to use either by filename or tag. If the tag is ambiguous, then the most current copy is used (the one that requires the least media recovery).

If you do not specify the target of the switch, then the filename specified in a prior **set newname** for this file number is used as the switch target. If you specify **switch datafile all**, then all datafiles for which a **set newname** has been issued in this job are switched to their new name.

Recovering Datafiles

You can use the Recovery Manager **recover** command to perform media recovery and apply incremental backups. Only current datafiles may be recovered or have incremental backups applied to them. Archive log backup sets are restored as needed to perform the media recovery. By default, the logs are restored to the current log archive destination as specified in the init.ora file. An operand is provided for specifying a different location.

If Recovery Manager has a choice between applying an incremental backup or applying redo, then it always chooses to use the incremental backup. If overlapping levels of incremental backup are available, then the lowest level of incremental backup (the one covering the longest period of time) is chosen automatically.

Guidelines for Recovering Datafiles

If possible, the recovery catalog should be made available to perform the recovery. If it is not available, then Recovery Manager uses information from the target database control file to perform the recovery if possible. Note that if control file recovery is required, then the recovery catalog must be available. Recovery Manager cannot operate when neither the recovery catalog nor the target database controlfile are available.

At least one **allocate channel** command must precede the **recover** command unless it is known that no archive log or incremental datafile backup sets will need restoration. Furthermore, the appropriate type of device(s) must be allocated for the backup sets that will need restoration. If the appropriate type of device is not available, then the **recover** command will fail. The restores of datafile incremental backup sets are performed in parallel if multiple channels are allocated, with the degree of parallelism controlled by the number of **allocate channel** commands.

Database Point-In-Time Recovery

An easy way of specifying the time to is to use **set until**. This command affects any *subsequent* **restore**, **switch** and **recover** commands in the same **run** command. If a **set until** command is specified after a **restore** and before a **recover**, you may not be able to recover the database to the point in time required, as the files restored may already have timestamps more recent than that time; for this reason, the **set until** command is usually specified before the **restore** or **switch** command.

set until

This specifies a point in time for a subsequent restore or recover command. The point in time may be specified using one of the following keywords:

- time *string*

- logseq *integer* **thread** *integer*

- scn *integer*

If the **time** keyword is specified then *string* must be a formatted according to the NLS date format specification currently in effect. This format is specified by the NLS_DATE_FORMAT environment variable on most platforms.

Recovery Commands

There are three forms of Recovery Manager **recover** commands:

- **recover database**
- **recover tablespace**
- **recover datafile**

For datafile and tablespace recovery, the target database must be started and mounted. If it is open, then the datafile(s) or tablespace(s) to be recovered must be offline.

For database recovery, the database must be started but not open. If possible, the target database should be mounted. If the target database is not mounted, then Recovery Manager uses the values of the **db_name** and **db_domain** init.ora parameters to determine which target database it is operating against. If the **db_name** parameter is not specified in the target database init.ora file, then the **recover** command will fail. If there are multiple target databases with the same name and domain in the recovery catalog, again, the **recover** command will fail.

Datafile Recovery

Perform this form of recovery when a media failure has damaged one or more datafiles.

Tablespace Recovery

Perform this form of recovery when a media failure has damaged all datafiles for a tablespace, or when a tablespace is to be recovered to a previous point-in-time.

Performing tablespace recovery is also an easy way of naming the files that are to be recovered. Any files not requiring recovery are simply ignored.

Database Recovery

Perform database recovery under the following circumstances:

- media failure has damaged the entire database
- the entire database is to be recovered to a previous point-in-time
- media failure has damaged the control file
- **create controlfile** has been executed

Recover Command Object List

datafile

This specifies a list of one or more datafiles to recover. Datafiles may be specified by filename or filenumber. The name must be the current name of the datafile as known in the recovery catalog. If the datafile has been renamed in the control file since the last time it was backed up or the last time a **resync catalog** was performed, then the old name of the datafile must be used.

tablespace

This specifies a list of one or more tablespaces to recover.

database

This specifies that the entire database is to be recovered. You can specify an optional *until_clause* that causes the recovery to stop when the specified until condition has been reached.

An optional ***skip*** clause may also be specified. The ***skip*** clause lists tablespaces that should not be recovered. This is useful for avoiding recovery of tablespaces containing only temporary data, or for postponing recovery of some tablespaces until a later time.

The **skip** clause causes Recovery Manager to take the datafiles belonging to the specified tablespaces offline before starting media recovery. These files are left offline after the media recovery is complete. If an incomplete recovery is being performed, then **skip** is not allowed. Instead, you must use **skip forever**, with the intention of dropping the skipped tablespaces after opening the database with the RESETLOGS option.

The **skip forever** clause causes Recovery Manager to take the datafiles offline using the **drop** option. Only use **skip forever** when the specified tablespaces will be dropped after opening the database.

Monitoring Backups and Restores

There are a number of ways to identify what a server process performing a backup, restore or copy is doing. This section discusses two options.

Connecting a Session to Channels

To identify which server processes correspond to which Recovery Manager channels, use **set command id** and query the V$SESSION.CLIENT_INFO column.

set command id

This enters the specified *string* into the V$SESSION.CLIENT_INFO column of all channels. This can be used to identify which Oracle server processes correspond to which channels.

The V$SESSION.CLIENT_INFO column will contain information for each Recovery Manager server process, in one of the following formats:

- **id=*string***

  This form appears for the first connection to the target database established by Recovery Manager.

- **id=***string*, ch=*channel_id*

  This form appears for all allocated channels.

Example: Using set command id

```
run {
  set command id to 'rman';
  allocate channel t1 type 'SBT_TAPE';
  allocate channel t2 type 'SBT_TAPE';
  backup
    incremental level 0
    format 'df_%t_%s_%p'
    filesperset 5
   (tablespace data_1);
 sql 'alter system archive log all';
}
```

Monitoring Progress

You can monitor the progress of backups, copies, and restores by querying the view V$SESSION_LONGOPS.

Each server process performing a backup, restore or copy reports its progress compared to the total amount of work to do for that particular part of the restore.

For example, if a restore was being performed using two channels, and each channel had two backup sets to restore (a total of 4 sets), each server process would update report its progress through a single set. When that set was completely restored, it would then start reporting progress on the next set to restore.

The information can be queried using the following SQL statement:

```
SELECT sid, serial#, context, sofar, totalwork
       round(sofar/totalwork*100,2) "% Complete",
  FROM v$session_longops
  WHERE compnam = 'dbms_backup_restore';
```

9

Recovery Manager Scenarios

This chapter provides scenarios that use Recovery Manager, and includes the following topics:

- Backing Up in NOARCHIVELOG Mode
- Backing Up Databases and Tablespaces
- Backing Up in a Parallel Server Environment
- Copying Datafiles
- Incremental Backups
- Handling Errors
- Using O/S Utilities To Make Copies
- Keeping Backups
- Restoring and Recovering
- Database Point-In-Time Recovery
- Querying the Recovery Catalog
- Using the Report Command for Complex Recovery Catalog Queries

Backing Up in NOARCHIVELOG Mode

In this scenario the database is operating in NOARCHIVELOG mode, and the administrator has chosen to shut it down cleanly and back it up.

It is possible to skip tablespaces, but any skipped tablespace that has not been offline or read-only since its last backup will be lost if the database has to be restored from a backup.

You must use Server Manager/LineMode to shutdown the database.

```
SVRMGR> shutdown;
SVRMGR> startup mount
```

Then start Recovery Manager and enter the following:

```
run {
# backup the database to disk
allocate channel dev1 type disk;
backup (database format '/oracle/backups/bp_%s_%p'); }
```

The filename of the backup piece is generated using the format string. When the backup is written to disk, it is important to make sure that the destination (file system or raw device) has enough free space.

Backing Up Databases and Tablespaces

In the following examples, the database is running in ARCHIVELOG mode.

Backing Up a Database

To back up a database to tape you must first allocate a tape device. Query V$BACKUP_DEVICE to see what devices are available to you.

The following example shows how to back up the database (except tablespaces that are offline):

```
run {
allocate channel dev1 type 'sbt_tape';
backup skip offline (database format '%d_%u'); }
```

A read-only tablespace needs to be backed up only once or twice after it has been made read-only. You can use the **skip read only** option to skip read-only datafiles. If you use the **skip offline** option, then the **backup** will not attempt to access offline datafiles. Use this option if the offline datafiles are not available.

Backing Up a Tablespace

The following example shows how to back up individual tablespaces. It is important to back up tablespaces that contain important data frequently (including system data and any tablespace that contains rollback segments). Tablespaces containing only temporary segments need not be backed up. Because this example backs up to disk, the format string determines the name of the backup file.

```
run {
allocate channel dev1 type disk;
backup
(tablespace system,tbs_1,tbs_2,tbs_3,tbs_4,tbs_5
format '/oracle/backups/bp_%s_%p');
}
```

Backing Up Individual Datafiles

You can also use Recovery Manager to back up individual datafiles as follows:

```
run {
allocate channel dev1 type 'sbt_tape';
backup
(datafile '?/dbs/t_dbs1.f'
format '%d_%u');
}
```

Backing Up the Control File

The current control file is automatically backed up when the first datafile of the system tablespace is backed up. The current control file can also be explicitly included in a backup or backed up individually.

```
run {
allocate channel dev1 type 'sbt_tape';
backup
(tablespace tbs_5 include current controlfile
format '%d_%u');
}
```

Backing Up Archived Logs

You can also back up archived logs to tape. The range of archived logs can be specified by time or log sequence. Note that specifying an archive log range does not guarantee that all redo in the range is backed up. For example, the last archived log may end before the end of the range, or an archived log in the range may be

missing. Recovery Manager simply backs up the logs it finds and does not issue a warning. Note that online logs cannot be backed up; they must be archived first.

> **Hint:** Set the NLS_LANG and NLS_DATE_FORMAT environment variables before invoking Recovery Manager.

```
 NLS_LANG=american
NLS_DATE_FORMAT='Mon DD YYYY HH24:MI:SS'

run {
allocate channel dev1 type 'sbt_tape';
backup
(archivelog  from time  'Nov 13 1996 20:57:13'
             until time 'Nov 13 1996 21:06:05'
all
format '%d_%u');
}
```

Here we back up all archived logs from sequence# 288 to sequence# 301 and delete the archived logs after the backup is complete. If the backup fails the logs are not deleted.

```
run {
allocate channel dev1 type 'sbt_tape';
backup
(archivelog low logseq 288 high logseq 301 thread 1
all delete input
format '%d_%u');
}
```

The following commands back up all archived logs generated during the last 24 hours. We archive the current log first to ensure that all redo generated up to the present gets backed up.

```
run {
allocate channel dev1 type 'sbt_tape';
sql "alter system archive log current";
backup
(archivelog  from time 'SYSDATE-1' all
format '%d_%u') ;
}
```

See Also: For more information about your environment variables, see your operating system-specific documentation.

Backing Up in a Parallel Server Environment

The following script distributes datafile and archivelog backups across two instances in a parallel server environment:

```
run {

allocate channel node_1 type disk connect 'internal/kernel@node_1';
allocate channel node_2 type disk connect 'internal/kernel@node_2';
   backup
      filesperset 1
      format 'df_%s_%p'
      (tablespace system, rbs, data1, data2
         channel node_1)
      (tablespace temp, reccat, data3, data4
         channel node_2);
   backup
      filesperset 20
      format 'al_%s_%p'
      (archivelog
         until time 'SYSDATE'
         thread 1
         delete input
            channel node_1);
       (archivelog
            until time 'SYSDATE'
            thread 2
            delete input
            channel node_2);
release channel node_1;
release channel node_2;
}
```

Copying Datafiles

Here we use Recovery Manager to make copies of datafiles to disk. A datafile copy is an image copy of the datafile.

```
run {
allocate channel dev1 type disk;
copy datafile '?/dbs/tbs_01.f/dbs/tbs_01.f' to '?/copy/temp3.f';
}
```

Incremental Backups

An incremental backup contains only blocks that have been changed since the previous backup. The first incremental backup must be a level 0 backup that contains all used blocks.

```
run {
allocate channel dev1 type 'sbt_tape';
backup incremental level 0
(database
format '%d_%u');
}
```

The next incremental backup at level 1 or higher will contain all blocks changed since the previous level 0 or level 1 backup.

```
run {
allocate channel dev1 type 'sbt_tape';
backup incremental level 1
(database
format '&d_%u');
}
```

If a new datafile or tablespace is added to the database then make a level 0 backup before the incremental backup. Otherwise the incremental backup of the tablespace or the database will fail because Recovery Manager doesn't find a parent backup for the new datafiles.

```
run {
allocate channel dev1 type 'sbt_tape';
backup incremental level 0
(tablespace new_tbs
format '%d_%u');
}
```

You can perform incremental backups in NOARCHIVELOG mode.

Handling Errors

By default a checksum is calculated for every block read from a datafile and stored in the backup. If you use the **nochecksum** option then checksums are not calculated. However, if the block already contains a checksum, the checksum is validated and stored in the backup. If the validation fails, the block is marked corrupt in the backup.

The **set maxcorrupt** clause determines how many corrupt blocks **backup** tolerates. If any datafile has more corrupt blocks than specified by **maxcorrupt**, the backup will terminate.

By default, **backup** will terminate if it cannot access a datafile. If the **skip inaccessible** option is specified, then **backup** will skip inaccessible datafiles and continue to back up other files. If **skip offline** is used, then **backup** will not attempt to access offline files.

```
run {
allocate channel dev1 type 'sbt_tape';
set maxcorrupt 5;
backup (database format 'bp_%s_%p'); }
```

Using O/S Utilities To Make Copies

Datafile copies you make using O/S utilities can be catalogued in the recovery catalog. Note that only copies made to disk can be catalogued. Because the format of backup pieces is proprietary, O/S utilities cannot write backups that Recovery Manager can read.

The datafile copies must be made using Oracle7 techniques. If the database is open and the datafile is online, you must first issue ALTER TABLESPACE BEGIN BACKUP. The resultant backup can be cataloged:

```
catalog datafilecopy '?/dbs/tbs_33.f';
```

Keeping Backups

How long backups and copies must be kept depends on factors such as:

- How frequently backups are taken
- How long the past point-in-time recovery is needed

For example, if all datafiles are backed up daily, no point-in-time recovery is needed and only one backup per datafile is sufficient, then previous backups can be deleted as soon as the new one completes.

```
# list all files####
# delete a specific datafilecopy
change datafilecopy '?/dbs/tbs_35.f' delete;

# delete archivelogs older than 31 days
change archivelog until time 'SYSDATE-31' delete';
```

There is currently no way to delete all the pieces of a multi-piece backup set with a single command. You must delete all the backup pieces individually.The deletion of any piece of a backup set automatically removes that backup set from consideration for restore and recovery.

You must allocate a channel before deleting a backup piece. The specified backup piece must have been created on the same type of device. Note that the **allocate channel for delete** command is *not* issued inside of a **run** command.

```
# delete a backup piece
allocate channel for delete type 'sbt_tape';
change backuppiece 'testdb_87fa39e0' delete;
release channel;
```

Restoring and Recovering

When and how to restore and/or recover information depends on the state of the database and the location of its datafiles.

The first step toward restoring/recovery is to determine the status of the database by executing the following query:

```
select status, parallel from v$instance;
```

Restore and Recover When the Database Is Open

If the status is open, then the database is open, but it is possible that some tablespaces and their datafiles need to be restored or recovered.

To recover or restore, execute the following query:

```
select file#, status, error, recover, tablespace_name, name
   from v$datafile_header;
```

If the error column is not null, then either the datafile could not be accessed or its header could not be validated. Unless the error is caused by a temporary hardware or operating system problem, the datafile must be restored or switched to a copy of that datafile:

```
run {
# recover tablespace tbs_1 while the database is open
allocate channel dev1 type 'sbt_tape';
sql "alter tablespace tbs_1 offline immediate" ;
restore tablespace tbs_1 ;
recover tablespace tbs_1 ;
sql "alter tablespace tbs_1 online" ;
release channel dev1 ;
}
```

If a datafile cannot be accessed due to a disk failure, it is likely that it must be restored to a new location or switched to an existing datafile copy. The following restore example allocates one disk channel and one 'sbt_tape' channel to allow restore to use datafile copies on disk and backups on 'sbt_tape'. This example allocates one disk device and one tape device to allow Recovery Manager to restore both from disk and tape.

```
run {
   allocate channel dev1 type disk;
   allocate channel dev2 type 'sbt_tape';
      sql "alter tablespace tbs_1 offline immediate" ;
      set newname for datafile 'disk7/oracle/tbs11.f'
      to 'disk9/oracle/tbs11.f' ;
      restore (tablespace tbs_1) ;
      switch datafile all ;
      recover tablespace tbs_1 ;
      sql "alter tablespace tbs_1 online" ;
   release channel dev1;
   release channel dev2;
}
```

Because V$DATAFILE_HEADER only reads the header block of each datafile it does not detect all problems that require the datafile to be restored. For example, if the datafile contains unreadable data blocks, but its header block is intact, no error is reported.

If the file can be accessed (error is null), but the recover column returns “yes,” then the file should be recovered. The Recovery Manager **recover** command first applies any suitable incremental backups and then applies archived logs and/or online logs. The incremental backups and archived logs are restored as needed.

```
run {
# recover tablespace tbs_1 while the database is open
   allocate channel dev1 type disk ;
      sql "alter tablespace tbs_1 offline" ;
      recover tablespace tbs_1 ;
      sql "alter tablespace tbs_1 online" ;
   release channel dev1 ;
}
```

Restore

If the database status is “mounted,” check the control file type:

```
select LOG_MODE, CONTROLFILE_TYPE, OPEN_RESETLOGS from v$database;
```

If the database status is “current,” try to open the database:

```
alter database open;
```

If the open succeeds, then use the procedure above to find out if any tablespaces and datafiles need recovery.

If the open fails, then you need to restore and / or recover the database first. If the control file type is “backup,” the database is probably being restored and recovered.

Database Point-In-Time Recovery

The database must be closed to perform database point-in-time recovery. You can use Server Manager to close the database:

```
SVRMGR> shutdown abort;
SVRMGR> startup nomount;
```

In this scenario, we are recovering the database up until 3 p.m. Assume that the disk containing tablespace TBS_1 datafiles crashed and that there is no backup of tablespace TEMP1 because it only contains temporary segments, no user data. A new location is specified for TBS_1 datafiles. Note that, by default Recovery Manager will restore the files to their most recent location, not to where they were located at 3 p.m.

Hint: Set the NLS_LANG and NLS_DATE_FORMAT environment variables before invoking Recovery Manager.

```
 NLS_LANG=american
NLS_DATE_FORMAT='Mon DD YYYY HH24:MI:SS'

run {
# recover database until 3pm after restoring tbs_1 to a new location
allocate channel dev1 type disk;
allocate channel dev2 type 'sbt_tape';
set until time 'Nov 15 1996 15:00:00'
set newname for datafile '/vobs/oracle/dbs/tbs_11.f'
to '?/dbs/temp1.f' ;
set newname for datafile '?/dbs/tbs_12.f'
to '?/dbs/temp2.f' ;
```

```
restore controlfile to '/vobs/oracle/dbs/cf1.f' ;
replicate controlfile from '/vobs/oracle/dbs/cf1.f';
sql "alter database mount" ;
restore database skip tablespace temp1;
switch datafile all;
recover database skip tablespace temp1;
sql "alter database open resetlogs";
sql "drop tablespace temp1";
sql "create tablespace temp1 datafile '/vobs/oracle/dbs/temp1.f' size 10M";
release channel dev1;
release channel dev2;
}
```

Now assume that log sequence 1234 was lost due to a disk crash and the database needs to be recovered using available archivelogs. We know that tablespace READONLY1 has not changed since logseq 1234, so we don't restore it.

```
run {
# recover database until log sequence 1234
allocate channel dev1 type disk;
allocate channel dev2 type 'sbt_tape';
set until logseq 1234 thread 1;
restore controlfile to '/vobs/oracle/dbs/cf1.f' ;
replicate controlfile from '/vobs/oracle/dbs/cf1.f';
sql "alter database mount" ;
restore database skip tablespace temp1, readonly1;
recover database skip tablespace temp1;
sql "alter database open resetlogs";
sql “drop tablespace temp1”;
sql "create tablespace temp1 datafile '/vobs/oracle/dbs/temp1.f' size 10M";
release channel dev1;
release channel dev2; }
```

Querying the Recovery Catalog

You can use the **list** command to query the contents of the recovery catalog, or the target database control file if no recovery catalog is used.

```
# list all backups of files in tablespace tbs_1 that were made since
# November first.
list until time 'Nov 1 1996 00:00:00' backupset of tablespace tbs_1;
# list all backups on device type 'sbt_tape'
list device type 'sbt_tape' backupset of database;
# list all copies of a datafile, qualified by tag and directory.
list tag foo like '/somedir/%' copy of datafile 21;
# list all database incarnations registered in the recovery catalog
list incarnation of database;
```

You can also use the **list** command to determine which copies and backups can be deleted. For example, if a full backup of the database was created on November 2, and it will not be necessary to recover the database to an earlier point-in-time, then the backup sets listed in the following report can be deleted:

```
list until time 'Nov 1 1996 00:00:00' backupset of database;
```

Using the Report Command for Complex Recovery Catalog Queries

You can use the **report** command to issue more complex queries against the recovery catalog. The **report** command also works with the target database control file if no recovery catalog is used.

```
# report on all datafiles which need a new backup because they contain
# unlogged changes that were made after the last full or incremental backup.
report unrecoverable database;
# report on all datafiles which need a new backup because 3 or more incremental
# backups have been taken since the last full backup.
report need backup incremental 3 database;

# report on all datafiles in tablespace tbs_1 which need a new backup because
# the last full or incremental backup was taken more than 5 days ago.
report need backup days 5 database;
```

10

Recovery Manager Tablespace Point-in-Time Recovery

Attention: Due to the complex nature of tablespace point-in-time recovery, Oracle recommends that you contact and work with Worldwide Customer Support Services before using the procedures described here.

This chapter describes how to use Recovery Manager (RMAN) to perform tablespace point-in-time recovery (TSPITR), and includes the following topics:

- Introduction to Recovery Manager Tablespace Point-in-Time Recovery
- Planning for Recovery Manager Tablespace Point-in-Time Recovery
- Performing Recovery Manager Tablespace Point-In-Time Recovery
- Tuning Considerations

Introduction to Recovery Manager Tablespace Point-in-Time Recovery

Recovery Manager (RMAN) automated Tablespace Point-In-Time Recovery (TSPITR) enables you to quickly recover one or more tablespaces to a point-in-time that is different from that of the rest of the database. TSPITR is most useful in the following situations:

- To recover from an erroneous drop or truncate table operation.
- To recover a table that has become logically corrupted.
- To recover from an incorrect batch job or other DML statement that has affected only a subset of the database.
- In cases where there are multiple logical databases in separate tablespaces of one physical database, and where one logical database must be recovered to a point different from that of the rest of the physical database.
- For VLDBs (very large databases) even if a full database point-in-time recovery would suffice, you might choose to do tablespace point-in-time recovery rather than restore the whole database from a backup and perform a complete database roll-forward (see "Planning for Recovery Manager Tablespace Point-in-Time Recovery" on page 10-3 before making any decisions).

Similar to a table export, RMAN TSPITR enables you to recover a consistent data set; however, the data set is the entire tablespace rather than just one object.

Prior to Oracle8, point-in-time recovery could only be used on a subset of a database by:

1. Creating a copy of the database
2. Rolling the copied database forward to the desired point in time
3. Exporting the desired objects from the copied database
4. Dropping the relevant objects from the production database
5. Importing the objects into the production database

However, there was a performance overhead associated with exporting and importing large objects.

Familiarize yourself with the following terms and abbreviations, which are used throughout this chapter:

TSPITR
Tablespace Point-in-Time Recovery

Clone Database

The copied database used for recovery in Oracle 8 TSPITR is called a "clone database", and has various substantive differences from a regular database.

Recovery Set

Tablespaces that require point-in-time recovery to be performed on them.

Auxiliary Set

Any other items required for TSPITR, including:

- backup control file
- system tablespaces
- datafiles containing rollback segments
- temporary tablespace (optional)

A small amount of space is required by export for sort operations. If a copy of the temporary tablespace is not included in the auxiliary set, then you must provide sort space either by creating a new temporary tablespace after the clone has been started up, or by setting autoextend to ON on the system tablespace files.

Planning for Recovery Manager Tablespace Point-in-Time Recovery

Recovery Manager TSPITR is a complicated procedure and requires careful planning. Before proceeding you should read all of this chapter thoroughly.

WARNING: You should not perform RMAN TSPITR for the first time on a production system, or during circumstances where there is a time constraint.

Attention: Many of the limitations and planning steps in this chapter can also be found in Chapter 13, “Performing Tablespace Point-in-Time Recovery”; however, differences in limitations and planning exist. These differences are explicitly called to your attention in this chapter.

Limitations

This section describes the limitations associated with performing RMAN TSPITR. As you begin the planning stage, familiarize yourself with these limitations.

The primary issue you should consider when deciding whether or not to perform RMAN TSPITR is the possibility of application-level inconsistencies between tables in recovered and unrecovered tablespaces (due to implicit rather than explicit referential dependencies). You must understand these dependencies, and also have the means to resolve any possible inconsistencies before proceeding.

Recovery Manager Only Supports Recovery Sets Containing Whole Tables

Note: This limitation is specific to RMAN TSPITR.

Recovery Manager TSPITR only supports recovery sets that contain whole tables. If, for example, you are performing RMAN TSPITR on partitioned tables, and have partitions spread across more than one table, RMAN will return an error message during the export phase of TSPITR. Recovery sets that contain tables without their constraints, or only the constraints without the table, will also result in errors.

Recovery Manager Does Not Support Tables Containing Rollback Segments

Note: This limitation is specific to RMAN TSPITR.

If you are performing *manual* TSPITR, you can take rollback segments in the recovery set offline—thus preventing changes being made to the recovery set before recovery is complete. However, RMAN TSPITR does not support recovery of tablespaces containing rollback segments. For more information about TSPITR and rollback segments, see “Step 3: Prepare the Primary Database for TSPITR” on page 13-12.

TS_PITR_CHECK Does Not Check for Objects Owned by SYS

The TS_PITR_CHECK view provides information on dependencies and restrictions that can prevent TSPITR from proceeding. However, TS_PITR_CHECK does not provide information about dependencies and restrictions for objects owned by SYS.

If there are any objects, including undo segments, owned by SYS in the recovery set, there is no guarantee that you can successfully recover these objects (because TSPITR utilizes the Export and Import utilities, which do not operate on objects

owned by SYS). To find out which recovery set objects are owned by SYS, issue the following statement:

```
SELECT OBJECT_NAME, OBJECT_TYPE
   FROM SYS.DBA_OBJECTS
   WHERE TABLESPACE_NAME IN ('<tablespacename1>','<tablespacename','
   <tablespace name  N')   and owner = 'SYS';
```

See Also: For more details about the TS_PITR_CHECK view, see "Step 2: Research and Resolve Dependencies on the Primary Database" on page 13-8.

TS_PITR_CHECK Does Not Detect Snapshot Tables

The TS_PITR_CHECK view does not detect snapshot tables (it does detect snapshot logs); they are exported as stand-alone tables. Thus, if a snapshot is dropped at time 3, and a backup from time 1 is used to roll forward to time 2, after TSPITR is complete the snapshot table will have been created as a stand-alone table, but without its associated snapshot view.

Partitioned Tables and TS_PITR_CHECK

If any of the tablespaces supplied to the predicate contain the first segment of a partitioned table, then the result set of the TS_PITR_CHECK view is inverted. If tablespaces supplied to the predicate do not include the first segment of a partitioned table, then one row is returned for the partition in question. If the tablespaces supplied to the predicate contain the first segment of a partitioned table, the results are inverted (for example, one row is returned for every tablespace containing partitions of that partitioned table, but not the tablespace that was supplied to the predicate). Returned rows indicate that there is a conflict that you must resolve by exchanging the partitions with stand-alone tables.

Bitmap Indexes

You must drop and re-create bitmap indexes after you complete TSPITR. If you don't, they will be unusable. If any bitmap indexes exist on the tables, imports will fail even if the bitmap indexes have been dropped from the primary database. An incorrect index segment will also be created, despite the failure, and you will have to drop and re-build the index.

Non-Partitioned Global Indexes

The TS_PITR_CHECK view does not detect non-partitioned global indexes of partitioned tables that are outside the recovery set. This is apparent when the view is queried manually and also during the export and import phase of TSPITR. After

TSPITR completes, the old index still exists on the recovered table, even though no errors are returned. You must drop and re-create the index.

> **Note:** Because the index is still valid, queries that use the index will return incorrect rows.

General Restrictions

In addition to the preceding limitations, RMAN TSPITR has the following restrictions:

- You cannot use RMAN TSPITR to recover dropped tablespaces.
- You cannot use RMAN TSPITR to recover a tablespace that has been dropped and re-created with the same name.
- You cannot use RMAN TSPITR to remove a datafile that has been added to the wrong tablespace. If the file was added after the point to which RMAN TSPITR is being performed, then the file will still be part of the tablespace (and will be empty) after RMAN TSPITR is complete.
- You cannot use DML statements on the clone database—the clone database is for recovery purposes only.
- After RMAN TSPITR is complete, you cannot use existing backups of the recovery set datafiles for recovery; instead, you must take fresh backups of the recovered files. If you attempt to recover using a backup taken prior to performing TSPITR, recovery will fail (with the error message ORA 1247, 00000, “database recovery through TSPITR of tablespace %s”).
- RMAN TSPITR does not recover optimizer statistics for objects that have had statistics calculated on them; statistics must be re-calculated after performing TSPITR.

- The following object types are not allowed within the RMAN TSPITR recovery set:
  - replicated master tables
  - tables with varray columns
  - tables with nested tables
  - tables with external Bfiles
  - snapshot logs
  - snapshot tables
  - objects owned by SYS (including rollback segments)

Data Consistency and Recovery Manager TSPITR

TSPITR provides views that can detect any data relationships between objects that are in the tablespaces being recovered and objects in the rest of the database. TSPITR will not successfully complete unless these relationships are managed, either by removing or suspending the relationship, or by including the related object within the recovery set.

See Also: For more information see "Step 2: Research and Resolve Dependencies on the Primary Database" on page 13-8, and "TS_PITR_CHECK Does Not Check for Objects Owned by SYS" on page 10-4.

Recovery Manager TSPITR Planning Requirements

You must satisfy the following requirements before performing RMAN TSPITR.

Create an Oracle Password File for the Clone Instance

For information about creating and maintaining Oracle password files, see "Password File Administration" on page 1-9 of the *Oracle8 Administrator's Guide*.

Set init.ora Parameters for Clone Instance

You must set the following parameters in the *init.ora* file used by the clone instance.

- CONTROL_FILES
- DB_FILE_NAME_CONVERT
- LOG_FILE_NAME_CONVERT

These settings will apply to the control file name, converted datafile names, and converted logfile names at the clone database.

Example of init.ora Parameter Settings for Clone Instance: Following are examples of the init.ora parameter settings you must make for the clone instance.

```
control_files=/oracle/clone/cf/clon_prod_cf.f
db_file_name_convert=("/oracle/prod/datafile","/oracle/clone/datafile")
log_file_name_convert=("/oracle/prod/redo_log","/oracle/clone/redo_log")
```

See Also: For details about DB_FILE_NAME_CONVERT, see "Tuning Considerations" on page 10-10.

Attention: After setting these parameters, ensure that you do not overwrite the init.ora settings for the production files at the target database.

Create a Clone Database

Before beginning RMAN TSPITR, you must have a clone database up (startup NOMOUNT) and running.

Start the Recovery Manager Command Line Interface

Use either of the following two methods to start the RMAN command line interface.

Connect To the Clone Instance, Target Instance and Recovery Catalog To connect to the clone instance, target instance and recovery catalog, you must supply the following information when starting up Recovery Manager:

```
rman target sys/<tsyspwd>@<target_str>
     rcvcat rman /<rmanpwd>@<rcvcat_str>
     clone sys/<csyspwd>@<clone_str>
```

Where:

<tsyspwd>	The "connect sys" password specified in the target database's orapwd file
<target_str>	The TNS alias for the target database
<rmanpwd>	The "connect rman" password specified in the recovery catalog's orapwd file
<rcvcat_str>	The TNS alias for the recovery catalog database

<csyspwd>	The "connect sys" password specified in the clone database's orapwd file.
<clone_str>	The TNS alias for the clone database.

Start Up the Recovery Manager Without the Connection to the Clone Instance You can start the Recovery Manager command line interface without a connection to the clone instance, and then use the **connect clone** Recovery Manager command to make the clone connection, as follows:

```
RMAN> connect clone sys/<syspwd>@<clone_str>;
```

Performing Recovery Manager Tablespace Point-In-Time Recovery

After you have completed all planning requirements you can perform RMAN TSPITR.

To perform RMAN TSPITR, issue the following commands (where **<tbslist>** is the list of tablespace names in the recovery set):

```
'allocate clone channel'
'recover tablespace <tbslist> until'
```

At least one clone channel must be allocated with the **allocate clone channel** command.

> **Reminder:** The tablespace recovery set should not contain the system tablespace or any tablespace with rollback segments.

The following example statement runs RMAN TSPITR:

```
RMAN>  run  {
       allocate clone channel dev1 type 'SBT_TAPE';
       recover tablespace tbs_2, tbs_3 until time 'Jan 10 1998 20:00:00';
       }
```

Back Up Tablespaces After Recovery Manager TSPITR Is Complete

The tablespaces in the recovery set will remain offline until after RMAN TSPITR completes successfully.

You must make backups of tablespaces in the recovery set before placing these tablespaces online; all previous backups of datafiles in the recovery set are no longer valid.

RMAN TSPITR is not repeatable after completing successfully. At this point, all tablespaces and datafiles in the recovery set are assigned a new SCN. However, when you make a backup after RMAN TSPITR, you can perform another RMAN TSPITR to an **until_time** after this backup set creation time.

The clone database is not usable after a successful RMAN TSPITR; therefore, you should release the memory using the SHUTDOWN ABORT statement.

Should RMAN TSPITR be unsuccessful, SHUTDOWN ABORT the clone instance and restart (using NOMOUNT) after identifying and correcting the error. For example, if there is a conflict between the target database and the converted filename, the user should shutdown the clone instance, correct the converted datafile name, restart (using NOMOUNT), and run RMAN TSPITR again.

Also, after a successful RMAN TSPITR you can remove the following:

- clone auxiliary set datafiles restored to temporary locations during RMAN TSPITR
- clone database control files
- clone database redo log files

Tuning Considerations

This section describes issues that can affect the performance of RMAN TSPITR.

Specify a New Name for Datafiles in Auxiliary Set Tablespaces

Recovery Manager restores and recovers all datafiles belonging to the tablespaces in the recovery set and auxiliary set at the clone instance (Remember that the auxiliary set includes the system tablespace plus all the tablespaces with rollback segments). You can specify a new name for any datafiles in the auxiliary set tablespace using the **set newname** Recovery Manager command. Recovery Manager will use this new name as the temporary location in which to restore and recover the datafile. This new name will also override the setting in the

DB_FILE_NAME_CONVERT init.ora parameter. You can specify a new name for any datafiles in recovery set tablespaces.

If you specify a new name for datafiles in the recovery set tablespace, the datafile(s) will replace the original datafile in the control file at the target database—so the new filename replaces the existing filename.

Recovery Manager does not check for conflicts between datafile names at the clone and target databases. Any conflicts will result in an RMAN error.

Set the Clone Name and Use a Datafile Copy for Recovery Manager TSPITR

Using a datafile copy on disk is much faster than restoring a datafile. Hence, you may wish to use an appropriate copy of a datafile in the recovery or auxiliary set, instead of restoring and recovering a datafile.

Recovery Manager will use a datafile copy if the following 2 conditions are met:

1. The datafile copy name is registered in the recovery catalog as the clone name of the corresponding datafile via the following command (where **<datafile>** is the datafile name or number, and **<clone_datafilename>** is the datafile clone name):

```
RMAN> set clonename for datafile <datafile> to <clone_datafilename>
```

2. The datafile copy was made before the recovery "until_time" using the following RMAN command (where **<datafile>** is the datafile name):

```
RMAN> run  {
      copy datafile <datafile> to clonename;
      ...}
```

Examples

The following commands are examples of the conditions required by Recovery Manager:

```
RMAN> set clonename for datafile '/oracle/prod/datafile_1_1.dbf'
          to '/oracle/prod_copy/datafile_1_1.dbf';

RMAN> run  {
           allocate channel dev1 type disk;
           copy datafile '/oracle/prod/datafile_1_1.dbf'
           to clonename;
           }
```

Recovery Manager will not use a datafile copy if you use **set newname** for the same datafile.

If Recovery Manager uses a datafile copy and TSPITR completes successfully, the **clone_datafilename** will be marked "deleted" in the recovery catalog. The original datafile at the target will be replaced by this datafile copy after RMAN TSPITR is complete.

Use the Converted Datafile Name

If neither a new name nor clone name is set for a datafile in an auxiliary set tablespace, Recovery Manager can use the converted filename specified in the clone database control file to perform the restore and recovery. Recovery Manager checks for conflicts between datafile names at the clone and target databases. Any conflicts result in error.

If neither a new name nor clone name is set for a datafile in a recovery set tablespace, or the file at the clone name is unusable, Recovery Manager uses the original location of the datafile.

Part III

Backup and Recovery—Operating System

11

Performing Operating System Backups

This chapter explains how to back up the data in an Oracle database, and includes the following topics:

- Performing Backups
- Recovering from an Inconsistent Backup
- Using the Export and Import Utilities for Supplemental Database Protection

See Also: This chapter contains several references to Oracle Enterprise Manager. For more information about performing specific tasks using Enterprise Manager / GUI or Server Manager / LineMode, see the *Oracle Enterprise Manager Administrator's Guide.*

Performing Backups

This section describes the various aspects of taking database backups, and includes the following topics:

- Listing Database Files Before Backup
- Performing Whole Database Backups
- Performing Tablespace, Datafile, Control File and Archivelog Backups
- Performing Control File Backups

Listing Database Files Before Backup

Before taking a whole database (or tablespace, datafile, control file or archivlog) backup, identify the files to be backed up. Obtain a list of datafiles by querying the V$DATAFILE view:

```
SELECT name FROM v$datafile;
```

Then obtain a list of online redo log files for a database using the query below:

```
SELECT member FROM v$logfile;
```

These queries list the datafiles and online redo log files of a database, respectively, according to the information in the current control file of the database.

Finally, obtain the names of the current control files of the database by issuing the following statement within Enterprise Manager:

```
SHOW PARAMETER control_files;
```

Whenever you take a control file backup (using the ALTER DATABASE command with the BACKUP CONTROLFILE TO 'filename' option), save a list of all datafiles and online redo log files with the control file backup. To obtain this list use the ALTER DATABASE command with the BACKUP CONTROLFILE TO TRACE option. By saving the control file backup with the output of the TO TRACE invocation, the database's physical structure at the time of the control file backup is clearly documented.

Performing Whole Database Backups

Take a whole database backup of all files that constitute a database after the database is shut down to system-wide use in normal priority. *A whole database backup taken while the database is open, after an instance crash or shutdown abort is inconsistent.* In such cases, the backup is not a consistent whole database backup because the files are inconsistent with respect to a current point-in-time.

Whole database backups do not require the database to be operated in a specific archiving mode. A whole database backup can be taken if a database is operating in either ARCHIVELOG or NOARCHIVELOG mode. If the database is in NOARCHIVELOG mode, then the backup must be consistent (meaning the database is shut down cleanly before the backup).

The set of backup files that result from a consistent whole database backup are consistent because all files correspond to the same point in time. If a database restore is necessary, these files can completely restore the database to an exact point in time. After restoring the backup files, you may perform additional recovery steps to recover the database to a more current time if the database is operated in ARCHIVELOG mode. Also, you can take inconsistent whole database backups if your database is in ARCHIVELOG mode.

> **WARNING: A backup control file created during a whole database backup should only be used with the other files taken in that backup, to restore the backup. It should not be used for complete or incomplete database recovery. Unless you are taking a consistent whole database backup, you should back up your control file using the ALTER DATABASE command with the BACKUP CONTROLFILE option.**

See also: For more information about backing up control files, see "Performing Control File Backups" on page 11-10.

Preparing to Take a Consistent Whole Database Backup

To guarantee that a database's datafiles are consistent, always shut down the database with normal or immediate priority before making a whole database backup. Never perform a whole database backup after an instance failure or after the database is shut down with abort priority (that is, using a SHUTDOWN ABORT statement) unless your database is in ARCHIVELOG mode. In this case, the datafiles are probably not consistent with respect to a specific point-in-time.

To Prepare for a Whole Database Backup

1. Shut down the database with normal or immediate priority.

   To make a whole database backup, all database files must be closed by shutting down the database. Do not make a whole database backup when the instance is aborted or stopped because of a failure. Reopen the database and shut it down cleanly first.

2. Back up all files that constitute the database.

   Use operating system commands or a backup utility to make backups of all datafiles, and a single control file of the database. Also back up the parameter files associated with the database.

   You can perform OS backups:

   - within Enterprise Manager, using the HOST command
   - outside Enterprise Manager, with the operating system commands or a backup utility

3. Restart the database.

   After you have finished backing up all datafiles and a single control file of the database, you can restart the database.

Verifying Backups

DB_VERIFY is an external command-line utility that performs a physical data structure integrity check on an offline database. Use DB_VERIFY primarily when you need to ensure that a backup database (or datafile) is valid before it is restored or as a diagnostic aid when you have encountered data corruption problems.

See Also: The name and location of DB_VERIFY is dependent on your operating system (for example, dbv on Sun / Sequent systems). For more information about making operating system backups of files, see your operating system-specific Oracle documentation.

For more information on DB_VERIFY, see the *Oracle8 Utilities* guide.

Performing Tablespace, Datafile, Control File or Archivelog Backups

Tablespace, datafile, control file and archivelog backups should only be taken (and in some cases *can* only be taken) if a database is operating in ARCHIVELOG mode. Such backups cannot be used to restore a database operating in NOARCHIVELOG mode.

Backing Up Online Tablespaces and Datafiles

You can back up all datafiles of an individual online tablespace or specific datafiles of an online tablespace while the database is open.

When you back up an individual datafile or online tablespace, Oracle stops recording the occurrence of checkpoints in the headers of the online datafiles being backed up. Oracle stops recording checkpoints as a direct result of issuing the ALTER TABLESPACE BEGIN BACKUP statement. This means the database is in *hot backup* mode, and that when a datafile is restored, it has "knowledge" of the most recent datafile checkpoint that occurred *before* the online tablespace backup, not any that occurred *during* it. As a result, Oracle asks for the appropriate set of redo log files to apply should recovery be needed. After the datafile copy is completed, Oracle advances the file header to the current database checkpoint. This is done after the ALTER TABLESPACE END BACKUP statement is executed.

To Back Up Online Tablespaces in an Open Database

1. Identify the datafiles.

   If you are backing up a specific datafile, use the fully specified filename of the datafile.

   Before beginning a backup on an entire tablespace, identify all of the tablespace's datafiles using the DBA_DATA_FILES data dictionary view. For example, assume that the USERS tablespace is to be backed up. To identify the USERS tablespace's datafile, you can query the DBA_DATA_FILES view:

```
SELECT tablespace_name, file_name
FROM sys.dba_data_files
   WHERE tablespace_name = 'USERS';

TABLESPACE_NAME                  FILE_NAME
-------------------------------  -----------------
USERS                              filename1
USERS                              filename2
```

   Here, *filename1* and *filename2* are fully specified filenames corresponding to the datafiles of the USERS tablespace.

2. Mark the beginning of the online tablespace backup.

   To prepare the datafiles of an online tablespace for backup (and put them in hot backup mode), use either the Start Online Backup menu item of Enterprise Manager, or the SQL command ALTER TABLESPACE with the BEGIN BACKUP option.

   The following statement marks the start of an online backup for the tablespace USERS:

   ```
   ALTER TABLESPACE users BEGIN BACKUP;
   ```

   WARNING: If you forget to mark the beginning of an online tablespace backup, or neglect to assure that the BEGIN BACKUP command has completed before backing up an online tablespace, the backup datafiles are not useful for subsequent recovery operations. Attempting to recover such a backup is a risky procedure, and can return errors that result in inconsistent data later. For example, the attempted recovery operation will issue a "fuzzy files" warning, and lead to an inconsistent database that will not open.

3. Back up the online datafiles.

   At this point, you can back up the online datafiles of the online tablespace from within Enterprise Manager, using the HOST command, by exiting Enterprise Manager and entering the operating system commands, or starting the Backup utility

4. Mark the end of the online tablespace backup.

   After backing up the datafiles of the online tablespace, indicate the end of the online backup using either the End Online Tablespace Backup dialog box of Enterprise Manager, or the SQL command ALTER TABLESPACE with the END BACKUP option.

   The following statement ends the online backup of the tablespace USERS:

   ```
   ALTER TABLESPACE users END BACKUP;
   ```

If you forget to indicate the end of an online tablespace backup, and an instance failure or SHUTDOWN ABORT occurs, Oracle assumes that media recovery (possibly requiring archived redo logs) is necessary at the next instance start up. To avoid

performing media recovery in this case, use the ALTER DATABASE datafile *filename* END BACKUP statement.

See Also: See the *Oracle8 Reference* for more information about the DBA_DATA_FILES data dictionary view.

See your operating system-specific Oracle documentation for more information about making operating system backups of files.

Determining Datafile Backup Status To view the backup status of a datafile, you can use the data dictionary table V$BACKUP. This table lists all online files and gives their backup status. It is most useful when the database is open. It is also useful immediately after a crash, because it shows the backup status of the files at the time of the crash. You can use this information to determine whether you have left tablespaces in backup mode.

> **Note:** V$BACKUP is not useful if the control file currently in use is a restored backup or a new control file created since the media failure occurred. A restored or re-created control file does not contain the information Oracle needs to fill V$BACKUP accurately. Also, if you have restored a backup of a file, that file's STATUS in V$BACKUP reflects the backup status of the older version of the file, not the most current version. Thus, this view might contain misleading information on restored files.

For example, the following query displays the current backup status of datafiles:

```
SELECT file#, status
    FROM v$backup;

FILE#        STATUS
---------   ---------
0011         INACTIVE
0012         INACTIVE
0013         ACTIVE
...
```

In the STATUS column, "INACTIVE" indicates that the file is not currently being backed up. "ACTIVE" indicates that the file is marked as currently being backed up.

Backing Up Several Online Tablespaces If you have to back up several online tablespaces, use either of the following procedures:

- Back up the online tablespaces in parallel. For example, prepare all online tablespaces for backup:

```
ALTER TABLESPACE ts1 BEGIN BACKUP;
ALTER TABLESPACE ts2 BEGIN BACKUP;
ALTER TABLESPACE ts3 BEGIN BACKUP;
```

Next, back up all files of the online tablespaces and indicate that the online backups have been completed:

```
ALTER TABLESPACE ts1 END BACKUP;
ALTER TABLESPACE ts2 END BACKUP;
ALTER TABLESPACE ts3 END BACKUP;
```

- Back up the online tablespaces serially. For example, individually prepare, back up, and end the backup of each online tablespace:

```
ALTER TABLESPACE ts1 BEGIN BACKUP;
backup files
ALTER TABLESPACE ts1 END BACKUP;
ALTER TABLESPACE ts2 BEGIN BACKUP;
backup files
ALTER TABLESPACE ts2 END BACKUP;
```

The second option minimizes the time between ALTER TABLESPACE... BEGIN / END BACKUP commands and is recommended. During online backups, more redo information is generated for the tablespace.

Backing Up Offline Tablespaces and Datafiles

All or some of the datafiles of an individual tablespace can be backed up while the tablespace is offline. All other tablespaces of the database can remain open and available for system-wide use.

> **Note:** You cannot take the SYSTEM tablespace or any tablespace with active rollback segments offline. The following procedure cannot be used for such tablespaces.

To take tablespaces offline and online, you must have the MANAGE TABLESPACE system privilege.

1. Identify the datafiles of the offline tablespace.

   Use the fully specified filename of the datafile.

   Before taking the tablespace offline, identify the names of its datafiles by querying the data dictionary view DBA_DATA_FILES. (See Step 1 on on page 11-4.)

2. Take the tablespace offline, using normal priority if possible.

   Use of normal priority, if possible, is recommended because it guarantees that the tablespace can be subsequently brought online without the requirement for tablespace recovery.

   To take a tablespace and all associated datafiles offline with normal priority, use the Take Offline menu item of Enterprise Manager, or the SQL command ALTER TABLESPACE with the OFFLINE parameter. The following statement takes a tablespace named USERS offline normally:

   ```
   ALTER TABLESPACE users OFFLINE NORMAL;
   ```

   After a tablespace is taken offline with normal priority, all datafiles of the tablespace are closed.

3. Back up the offline datafiles.

   At this point, you can back up the datafiles of the offline tablespace from within Enterprise Manager using the HOST command, by exiting Enterprise Manager and entering the operating system commands, or starting the Backup utility.

4. Bring the tablespace online. *(Optional)*

   Bring the tablespace online using either the Place Online menu item of Enterprise Manager, or the SQL command ALTER TABLESPACE with the ONLINE option. The following statement brings an offline tablespace named USERS online:

   ```
   ALTER TABLESPACE users ONLINE;
   ```

   Note: If you took the tablespace offline using temporary or immediate priority, the tablespace may not be brought online unless tablespace recovery is performed.

After a tablespace is brought online, the datafiles of the tablespace are open and available for use.

See Also: For more information about making operating system backups of files, see your operating system-specific Oracle documentation.

Performing Control File Backups

You should back up the control file of a database after making a structural modification to a database operating in ARCHIVELOG mode.

To back up a database's control file, you must have the ALTER DATABASE system privilege.

You can take a backup of a database's control file using the SQL command ALTER DATABASE with the BACKUP CONTROLFILE option. The following statement backs up a database's control file:

```
ALTER DATABASE BACKUP CONTROLFILE TO 'filename' REUSE;
```

Here, *filename* is a fully specified filename that indicates the name of the new control file backup.

The REUSE option allows you to have the new control file overwrite a control file that currently exists.

Backing Up the Control File to the Trace File

The TRACE option of the ALTER DATABASE BACKUP CONTROLFILE command helps you manage and recover your control file. TRACE prompts Oracle to write SQL commands to the database's trace file, rather than making a physical backup of the control file. These commands start up the database, re-create the control file, and recover and open the database appropriately, based on the current control file. Each command is Commented. Thus, you can copy the commands from the trace file into a script file, edit them as necessary, and use the script to recover the database if all copies of the control file are lost (or to change the size of the control file).

For example, assume the SALES database has three enabled threads, of which thread 2 is public and thread 3 is private. It also has multiplexed redo log files, and one offline and one online tablespace.

```
ALTER DATABASE
    BACKUP CONTROLFILE TO TRACE NORESETLOGS;

3-JUN-1992 17:54:47.27:
# The following commands will create a new control file and use it
# to open the database.
```

```
# No data other than log history will be lost. Additional logs may
# be required for media recovery of offline data files. Use this
# only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE SALES NORESETLOGS ARCHIVELOG
     MAXLOGFILES 32
     MAXLOGMEMBERS 2
     MAXDATAFILES 32
     MAXINSTANCES 16
     MAXLOGHISTORY 1600
LOGFILE
     GROUP 1
       '/diska/prod/sales/db/log1t1.dbf',
       '/diskb/prod/sales/db/log1t2.dbf'
     )  SIZE 100K
    GROUP 2
       '/diska/prod/sales/db/log2t1.dbf',
        '/diskb/prod/sales/db/log2t2.dbf'
    ) SIZE 100K,
    GROUP 3
       '/diska/prod/sales/db/log3t1.dbf',
       '/diskb/prod/sales/db/log3t2.dbf'
    ) SIZE 100K
DATAFILE
    '/diska/prod/sales/db/database1.dbf',
    '/diskb/prod/sales/db/filea.dbf'
;
# Take files offline to match current control file.
ALTER DATABASE DATAFILE '/diska/prod/sales/db/filea.dbf' OFFLINE

# Recovery is required if any data files are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE;

# All logs need archiving and a log switch is needed.
ALTER SYSTEM ARCHIVE LOG ALL;

# Database can now be opened normally
ALTER DATABASE OPEN;

#  Files in normal offline tablespaces are now named.
ALTER DATABASE RENAME FILE 'MISSING0002'
    TO '/diska/prod/sales/db/fileb.dbf';
```

Using the command without NORESETLOGS produces the same output. Using the command with RESETLOGS produces a similar script that includes commands that recover and open the database, but resets the redo logs upon startup.

Recovering From a Failed Online Tablespace Backup

The following situations can cause a tablespace backup to fail and be incomplete:

- You did not indicate the end of the online tablespace backup operation (using the ALTER TABLESPACE command with the END BACKUP option), and the database was subsequently shut down with the ABORT option.
- A system or instance failure, or SHUTDOWN ... ABORT interrupted the backup.

Upon detecting an incomplete online tablespace backup at startup, Oracle assumes that media recovery (possibly requiring archived redo log) is necessary for startup to proceed. You can avoid performing media recovery by using the ALTER DATABASE DATAFILE ... END BACKUP command. Remember to list all the datafiles of the tablespaces that were in the process of being backup up before the database was restarted. You can determine whether datafiles were in the process of being backed up by querying the V$BACKUP view.

> **WARNING:** Do not use ALTER DATABASE DATAFILE ... END BACKUP if you have restored any of the affected files from a backup.

After you have restarted your database, you can perform the recovery in either of two ways:

- Use the STARTUP RECOVER command to start and recover the database automatically.
- Start an instance, open and mount the database, and issue the statement RECOVER DATABASE.

The first method is easier because it prompts Oracle to perform recovery only if it is needed.

See Also: For information on recovering a database, see Chapter 12, “Recovering a Database”.

Using the Export and Import Utilities for Supplemental Database Protection

This section describes the Import and Export utilities, and includes the following topics:

- Using Import
- Using Export

Export and Import are utilities that move Oracle data in and out of Oracle databases. Export writes data from an Oracle database to an operating system file in a special format. Import reads Export files and restores the corresponding information into an existing database. Although Export and Import are designed for moving Oracle data, you can also use them to supplement backups of data.

See Also: Both the Export and Import utilities are described in detail in the *Oracle8 Utilities* guide.

Using Export

The Export utility allows you to backup your database while it is open and available for use. It writes a read-consistent view of the database's objects to an operating system file. System audit options are not exported.

WARNING: If you use Export to backup, all data must be exported in a logically consistent way so that the backup reflects a single point in time. No one should make changes to the database while the Export takes place. Ideally, you should run the database in restricted mode while you export the data, so no regular users can access the data.

Table 11–1 lists available export modes.

Table 11–1 Export Modes

Mode	Description
User	exports all objects owned by a user
Table	exports all or specific tables owned by a user
Full Database	exports all objects of the database

Following are descriptions of Export types:

Incremental Export	Only database data that has changed since the last incremental, cumulative, or complete export is exported. An incremental export exports the object's definition and all its data. Incremental exports are typically performed more often than cumulative or complete reports. For example, if tables A, B, and C exist, and only table A's information has been modified since the last incremental export, only table A is exported.
Cumulative Exports	Only database data that has been changed since the last cumulative or complete export is exported. Perform this type of export on a limited basis, such as once a week, to condense the information contained in numerous incremental exports. For example, if tables A, B, and C exist, and only table A's and table B's information has been modified since the last cumulative export, only the changes to tables A and B are exported.
Complete Exports	All database data is exported. Perform this type of export on a limited basis, such as once a month, to export all data contained in a database.

Using Import

The Import utility allows you to restore the database information held in previously created Export files. It is the complement utility to Export.

To recover a database using Export files and the Import utility:

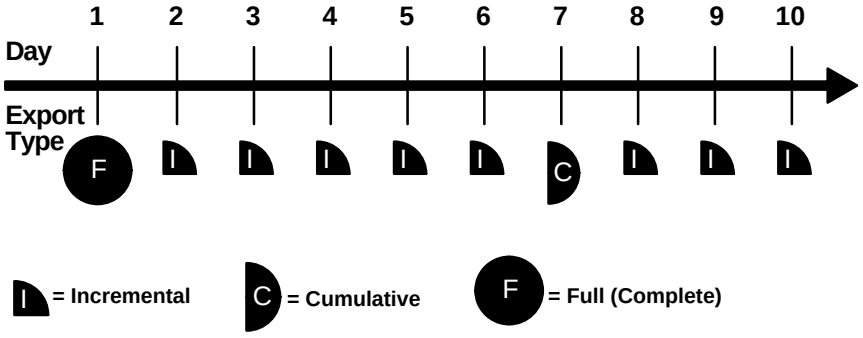
- Re-create the database structure, including all tablespaces and users.

  Note: These re-created structures should not have objects in them.

- Import the appropriate Export files to restore the database to the most current state possible. Depending on how your Export schedule is performed, imports of varying degrees will be necessary to restore a database.

- Assume that the schedule illustrated in Figure 11–1 is used in exporting data from an Oracle database

Figure 11–1 A Typical Export Schedule



A complete export was taken on Day 1, a cumulative export was taken every week, and incremental exports were taken daily.

1. Recreate the database, including all tablespaces and users.
2. Import the complete database export taken on Day 1.
3. Import the cumulative database export taken on Day 7.
4. Import the incremental database exports taken on Days 8, 9, and 10.

12

Recovering a Database

This chapter describes how to recover a database, and includes the following topics:

- Coordinate Distributed Recovery
- Recovery Scenarios
- Restoring a Whole Database Backup, NOARCHIVELOG Mode
- Specifying Parallel Recovery
- Preparing for Media Recovery
- Performing Complete Media Recoveryy
- Performing Incomplete Media Recovery
- Preparing for Disaster Recovery
- Unrecoverable Objects and Recovery
- Read-only Tablespaces and Recovery
- Recovery Procedure Examples

See Also: Occasionally, this chapter refers you to Oracle Enterprise Manager. To learn how to use Enterprise Manager / GUI or Server Manager / LineMode, see the *Oracle Enterprise Manager User's Guide*.

Coordinate Distributed Recovery

The Oracle distributed database architecture is autonomous in nature. Therefore, depending on the type of recovery operation selected for a single, damaged database, recovery operations may, or may not, have to be coordinated globally among all databases in the distributed database system. Table 12–1 summarizes the different types of recovery operations and whether coordination among nodes of a distributed database system is required.

Table 12–1 Recovery Operations in a Distributed Database Environment

Type of Recovery Operation	Implication for Distributed Database System
Restoring a whole backup for a database that was never accessed (updated or queried) from a remote node	Use non-coordinated, autonomous database recovery.
Restoring a whole backup for a database that was accessed by a remote node	Shut down all databases and restore them using the same coordinated full backup.
Complete media recovery of one or more databases in a distributed database	Use non-coordinated, autonomous database recovery.
Incomplete media recovery of a database that was never accessed by a remote node	Use non-coordinated, autonomous database recovery.
Incomplete media recovery of a database that was accessed by a remote node	Use coordinated, incomplete media recovery to the same global point-in-time for all databases in the distributed database.

Coordinate Time-Based and Change-Based Distributed Database Recovery

In special circumstances, one node in a distributed database may require recovery to a past point in time. To preserve global data consistency, it is often necessary to recover all other nodes in the system to the same point in time. This is called “coordinated, time-based, distributed database recovery.” The following tasks should be performed with the standard procedures of time-based and change-based recovery described in this chapter.

1. Recover the database that requires the recovery operation using time-based recovery. For example, if a database needs to be recovered because of a user error (such as an accidental table drop), recover this database first using time-based recovery. Do not recover the other databases at this point.
2. After you have recovered the database and opened it using the RESETLOGS option, look in the ALERT file of the database for the RESETLOGS message.

   If the message is, **"RESETLOGS after complete recovery through change nnnnnnnn,"** you have applied all the changes in the database and performed a complete recovery. Do not recover any of the other databases in the distributed system, or you will unnecessarily remove changes in them. Recovery is complete.

   If the reset message is, **"RESETLOGS after incomplete recovery UNTIL CHANGE nnnnnnnn,"** you have successfully performed an incomplete recovery. Record the change number from the message and proceed to the next step.
3. Recover all other databases in the distributed database system using change-based recovery, specifying the change number (SCN) from Step 2.

Recover Database with Snapshots

If a master database is independently recovered to a past point in time (that is, coordinated, time-based distributed database recovery is not performed), any dependent remote snapshot that was refreshed in the interval of lost time will be inconsistent with its master table. In this case, the administrator of the master database should instruct the remote administrators to perform a complete refresh of any inconsistent snapshot.

Recovery Scenarios

The following scenarios describe various ways to invoke media recovery.

Recovering a Closed Database

After the database is mounted, but closed, start closed database recovery (complete or incomplete) using either Enterprise Manager's Apply Recovery Archives dialog box, or the RECOVER command with the DATABASE parameter.

The following statement recovers the database up to a specified time using a control file backup:

```
RECOVER DATABASE
   UNTIL TIME '1992-12-31:12:47:30' USING BACKUP CONTROLFILE;
```

Recovering an Offline Tablespace in an Open Database

After the tablespaces of interest are taken offline, you can start open-database, offline-tablespace recovery using the RECOVER command with the TABLESPACE parameter. You can recover one or more offline tablespaces. The remainder of the database may be left open and online for normal database operation.

The following statement recovers two offline tablespaces:

```
RECOVER TABLESPACE ts1, ts2;
```

After the tablespaces that contain the damaged files have been taken offline, and you are positive the associated datafiles are also offline (check the file's status in V$DATAFILE), recover selected datafiles using the RECOVER command with the DATAFILE parameter:

```
RECOVER DATAFILE 'filename1', 'filename2';
```

Generally, you should perform database recovery using Enterprise Manager, which prompts you for information and returns messages from the system. The SQL command equivalent of Enterprise Manager media recovery options is the ALTER DATABASE command with the RECOVER clause. If you want to design your own recovery application using SQL commands, use the ALTER DATABASE command.

Starting Recovery During Instance Startup

You can start complete media recovery using the STARTUP command with the RECOVER option in Enterprise Manager. After an instance is started, and the database is mounted, complete media recovery proceeds as described in "Complete Media Recovery" on page "Performing Complete Media Recovery" on page 12-15.

See Also: For information about taking tablespaces offline, see "Backing Up Offline Tablespaces and Datafiles" on page 11-8.

Applying Redo Log Files

During complete or incomplete media recovery, redo log files (online and archived) are applied to the datafiles during the roll forward phase of media recovery. Because rollback data is recorded in the redo log, rolling forward regenerates the corresponding rollback segments. Rolling forward proceeds through as many redo log files as necessary to bring the database forward in time. As a log file is needed, Oracle suggests the name of the file. For example, if you are using Enterprise Manager, it returns the following lines and prompts:

```
ORA-00279: Change #### generated at DD/MM/YY HH:MM:SS needed for thread#
ORA-00289: Suggestion : logfile
ORA-00280: Change #### for thread # is in sequence #
Specify log: [<RET> for suggested | AUTO | FROM logsource |
    CANCEL ]
```

Similar messages are returned when using an ALTER DATABASE... RECOVER statement. However, no prompt is displayed.

Applying Log Files

This section describes how log files can be applied in different environments.

Suggested Log Filenames

Oracle suggests log filenames by concatenating the current values of the initialization parameters LOG_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT and using information from the control file. Thus, if all the required archived log files are mounted at LOG_ARCHIVE_DEST, and the value for LOG_ARCHIVE_FORMAT is never altered, Oracle can suggest and apply log files to complete media recovery automatically without your intervention. If the location specified by LOG_ARCHIVE_DEST is not available (for example, because of media failure), you can change the value for this parameter, move the log files to the new location, and start a new instance before beginning media recovery.

In some cases, you might want to override the current setting for LOG_ARCHIVE_DEST as a source for log files. For example, assume that a database is open and an offline tablespace must be recovered, but not enough space is available to mount the necessary log files at the location specified by LOG_ARCHIVE_DEST. In this case, you can mount the log files to an alternate location, then specify the alternate location to Oracle for the recovery operation. To specify the location where required log files can be found, use the LOGSOURCE parameter of the SET command in Enterprise Manager. Use the RECOVER...FROM parameter of the ALTER DATABASE command in SQL.

> **Note:** Overriding the log source does not affect the archive log destination for filled online groups being archived.

Consider overriding the current setting for LOG_ARCHIVE_DEST when not enough space is available to mount all the required log files at any one location. In this case, you can set the log file source to an operating system variable (such as a logical or an environment variable) that acts as a search path to several locations.

See Also: Such functionality is operating system-dependent. See your operating system-specific Oracle documentation for more information.

Applying Log Files when Using Enterprise Manager

If the suggested archived redo log file is correct, apply it. You do not have to specify a filename unless the suggested file is incorrect. After a filename is provided, Oracle applies the redo log file to roll forward the restored datafiles.

In Enterprise Manager, you can have Oracle automatically apply the redo log files that it suggests by choosing either of the following options:

- Before starting media recovery, issue the following Enterprise Manager statement to turn on automatic recovery:

  ```
  SET AUTORECOVERY ON;
  ```

  Automatic application of the suggested redo log starts once recovery begins.

- After media recovery is started, enter "auto" when prompted for a redo log file. Automatic application of the suggested redo log starts from this point.

Suggested redo log files are automatically applied until one is incorrect or recovery is complete. You might need to specify online redo log files manually when using cancel-based recovery or a backup of the control file.

See Also: For examples of logfile application, see your operating system-specific Oracle documentation.

Application of Log Files When Using SQL Commands

Application of redo log files is similar to the application of log files. However, a prompt for log files is not displayed after media recovery is started. Instead, you must provide the correct log file using an ALTER DATABASE RECOVER LOGFILE statement. For example, if a message suggests LOG1.ARC, you can apply the suggestion using the following statement:

```
ALTER DATABASE RECOVER LOGFILE 'log1.arc';
```

As a result, recovering a tablespace requires several statements, as indicated in the following example (DBA input is boldfaced; variable information is italicized.):

```
>ALTER DATABASE RECOVER TABLESPACE users;
ORA-00279: Change #### generated at DD/MM/YY HH:MM:SS needed for thread #
ORA-00289: Suggestion : logfile1
ORA-00280: Change #### for thread # is in sequence #
> ALTER DATABASE RECOVER LOGFILE 'logfile1';
ORA-00279: Change #### generated at DD/MM/YY HH:MM:SS needed for thread # <D%0>
```

```
ORA-00289: Suggestion : logfile2
ORA-00280: Change #### for thread # is in sequence #
> ALTER DATABASE RECOVER LOGFILE 'logfile2';
(Repeat until all logs are applied.)
Statement processed.
> ALTER TABLESPACE users ONLINE;
Statement processed.
```

In this example, assume that the backup files have been restored, and that the user has administrator privileges.

Like the method you used with Enterprise Manager, automatic application of the redo logs can be started with the following statements, before and during recovery, respectively:

```
ALTER DATABASE RECOVER AUTOMATIC ...;
ALTER DATABASE RECOVER AUTOMATIC LOGFILE suggested_log_file;
```

An example of the first statement follows:

```
> ALTER DATABASE RECOVER AUTOMATIC TABLESPACE users;
Statement processed.
> ALTER TABLESPACE users ONLINE;
Statement processed.
```

In this example, it is assumed that the backup files have been restored, and that the user has administrator privileges.

An example of the ALTER DATABASE RECOVER AUTOMATIC LOGFILE statement follows:

```
> ALTER DATABASE RECOVER TABLESPACE users;
ORA-00279: Change #### generated at DD/MM/YY HH:MM:SS needed for thread #
ORA-00289: Suggestion : logfile1
ORA-00280: Change #### for thread # is in sequence #
> ALTER DATABASE RECOVER AUTOMATIC LOGFILE 'logfile1';
Statement processed.
> ALTER TABLESPACE users ONLINE;
Statement processed.
```

In this example, assume that the backup files have been restored, and that the user has administrator privileges.

> **Note:** After issuing the ALTER DATABASE RECOVER command, you can view all files that have been considered for recovery in the V$RECOVERY_FILE_STATUS view. You can access status information for each file in the V$RECOVERY_STATUS view. These views are not accessible after you terminate the recovery session.

See Also: For information about the content of all recovery-related views, see the *Oracle8 Reference*.

Successful Application of Redo Logs

If you are using Enterprise Manager's recovery options (not SQL statements), each time Oracle finishes applying a redo log file, the following message is returned:

```
Log applied.
```

Make sure that the message "Log applied" is returned after each application of a redo log file. If the suggested file is incorrect or you provide an incorrect filename, an error message is returned instead. If you see an error message instead of "Log applied," a redo log file required for recovery has not been applied. Recovery cannot continue until the required redo log file is applied.

If an error message is returned after supplying a redo log filename, one of the following errors has been detected:

- If the error message says that the file cannot be found, you may have entered the wrong filename. Re-enter the correct filename.
- If the redo log file is found, but cannot be opened, then it may be locked. Unlock the redo log file and re-enter the filename.
- If a redo log file is found and opened, but cannot be read, an I/O error is returned. In this case, the redo log file may have been only partially written or may have been corrupted. If you can locate an uncorrupted or complete copy of the log, you can simply apply that copy; you do not need to restart recovery. Otherwise, if no other copy of the log exists and you know the time of the last valid redo entry, you can perform time-based or change-based recovery; in this case, you must restart recovery from the beginning, including restoring backups.

Interrupting Media Recovery

If you start a media recovery operation and must then interrupt it (for example, because a recovery operation must end for the night and resume the next morning), you can interrupt recovery at any time by taking either of the following actions:

- Enter the word “cancel” when prompted for a redo log file.
- Use your operating system’s interrupt signal (if you must abort when recovering an individual datafile, or when automated recovery is in progress).

After recovery is canceled, it must be completed before opening a database for normal operation. To resume recovery, restart it. Recovery resumes where it left off when it was canceled.

> **WARNING: There are several reasons why, after starting recovery, you may want to restart. If, for example, you want to restart with a different backup or want to use the same backup, but need to change the end-time to an earlier point in time than you initially specified, then *the entire operation must recommence by restoring a backup*. Failure to do so may result in “file inconsistent” error messages when attempting to open the database.**

Restoring a Whole Database Backup, NOARCHIVELOG Mode

If a database is in NOARCHIVELOG mode and a media failure damages some or all of the datafiles, usually the only option for recovering the database is to restore the most recent whole database backup. If you are using Export to supplement regular backups, you can instead restore the database by importing an exported backup of the database.

The disadvantage of NOARCHIVELOG mode is that to recover your database from the time of the most recent full backup up to the time of the media failure, you have to re-enter manually all of the changes executed in that interval. However, if your database was in ARCHIVELOG mode, the redo log covering this interval would have been available as archived log files or online log files. This would have enabled you to use complete or incomplete recovery to reconstruct your database and minimize the amount of lost work.

If you have a database damaged by media failure and operating in NOARCHIVELOG mode, and you want to restore from your most recent whole database backup (your only option at this point), perform the following tasks.

To Restore the Most Recent Whole Database Backup (NOARCHIVELOG Mode)

1. If the database is open, shut it down using the Enterprise Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. If the hardware problem that caused the media failure has been corrected so that the backup database files can be restored to their original locations, follow only Step 2a before proceeding to Step 3. If, on the other hand, the hardware problem has not been corrected and some or all of the database files must be restored to alternative locations, follow Steps 2a through 2d
   a. Restore the most recent whole database backup. All of the datafiles and control files of the whole database backup must be restored, not just the damaged files. This guarantees that the entire database is consistent to a single point in time.
   b. If necessary, edit the restored parameter file to indicate the new location of the control files.
   c. Start an instance using the restored and edited parameter file and mount, but do not open, the database.
   d. Perform the steps necessary to record the relocation of the restored datafiles. If applicable, perform the steps necessary to record the relocation of online redo log files.
3. Issue the RECOVER DATABASE UNTIL CANCEL command, which mimics incomplete database recovery.
4. Issue the ALTER DATABASE OPEN RESETLOGS command, which opens the database and resets the current log sequence to 1. It also invalidates all redo entries in the online redo log file. Restoring from a whole database backup and then resetting the log discards all changes to the database made from the time the backup was taken to the time of the media failure.

See Also: For more information about renaming and relocating datafiles, see the *Oracle8 Administrator's Guide*.

Specifying Parallel Recovery

The RECOVERY_PARALLELISM initialization parameter specifies the number of concurrent recovery processes to use for any recovery operation. Because crash recovery occurs at instance startup, this parameter is useful for specifying the number of processes to use for crash recovery. The value of this parameter is also the default number of processes used for media recovery if the PARALLEL clause of the RECOVER command is not specified. The value of this parameter must be greater than one and cannot exceed the value of the PARALLEL_MAX_SERVERS parameter.

In general, parallel recovery is most effective at reducing recovery time when several datafiles on several different disks are being recovered concurrently. Crash recovery (recovery after instance failure) and media recovery of many datafiles on different disk drives are good candidates for parallel recovery. Parallel recovery requires a minimum of eight recovery processes to improve upon serial recovery.

See Also: For more information on parallel recovery, see *Oracle8 Parallel Server Concepts and Administration*.

For more information about initialization parameters, see *Oracle8 Reference*.

Preparing for Media Recovery

This section describes issues related to media recovery preparation, and includes the following topics:

- Media Recovery Commands
- Issues Common to All Media Recovery Operations
- Restoring a Whole Database Backup, NOARCHIVELOG Mode
- Specifying Parallel Recovery

See Also: For information about the appropriate method of recovery for each type of problem, see “Recovery Procedure Examples” on page 12-45.

Media Recovery Commands

There are three basic media recovery commands, which differ only in the way the set of files being recovered is determined. They all use the same criteria for determining if files can be recovered. Media recovery signals an error if it cannot get the lock for a file it is attempting to recover. This prevents two recovery sessions from recovering the same file. It also prevents media recovery of a file that is in use.

You should be familiar with all media recovery commands before performing media recovery.

RECOVER DATABASE Command

RECOVER DATABASE performs media recovery on all online datafiles that require redo to be applied. If all instances were cleanly shutdown, and no backups were restored, RECOVER DATABASE indicates a no recovery required error. It also fails if any instances have the database open (since they have the datafile locks). To perform media recovery on an entire database (all tablespaces), the database must be mounted EXCLUSIVE and closed.

RECOVER TABLESPACE Command

RECOVER TABLESPACE performs media recovery on all datafiles in the tablespaces listed. The tablespaces must be offline to perform the recovery. An error is indicated if none of the files require recovery.

RECOVER DATAFILE Command

RECOVER DATAFILE lists the datafiles to be recovered. The database can be open or closed, provided the media recovery locks can be acquired. If the database is open in any instance, then datafile recovery can only recover offline files.

See Also: For more information about recovery commands, see the *Oracle8 SQL Reference*.

Issues Common to All Media Recovery Operations

This section describes topics common to all complete and incomplete media recovery operations. You should be familiar with these topics before proceeding with any recovery process.

Determining Which Files to Recover

You can often use the table V$RECOVER_FILE to determine which files to recover. This table lists all files that need to be recovered, and explains why they need to be recovered.

Note: The table is not useful if the control file currently in use is a restored backup or a new control file created since the media failure occurred. A restored or re-created control file does not contain the information Oracle needs to fill V$RECOVER_FILE accurately.

The following query displays the file ID numbers of datafiles that require recovery:

```
SELECT file#, online, error
    FROM v$recover_file;

FILE#      ONLINE     ERROR
--------   --------   --------------------------
   0014    ONLINE
   0018    ONLINE     FILE NOT FOUND
   0032    OFFLINE    OFFLINE NORMAL
...
```

Use the data dictionary view V$DATAFILE, which contains the file's NAME and FILE#, to find the name of a file based on its file number.

Restoring Damaged Datafiles

If a media failure permanently damages one or more datafiles of a database, you must restore backups of the damaged datafiles before you can recover the damaged files.

Relocating Damaged Files

If a damaged datafile cannot be restored to its original location (for example, a disk must be replaced, so the files are restored to an alternate disk), the new locations of these files must be indicated to the control file of the associated database.

Recovering a Datafile Without a Backup

If a datafile is damaged and no backup of the file is available, the datafile can still be recovered if:

- all log files written since the creation of the original datafile are available
- the control file contains the name of the damaged file (that is, the control file is current, or is a backup taken after the damaged datafile was added to the database

Use the CREATE DATAFILE clause of the ALTER DATABASE command to create a new, empty datafile, replacing a damaged datafile that has no corresponding backup. However, you cannot create a new file based on the first datafile of the SYSTEM tablespace because it contains information not covered by redo logs. For example, assume that the datafile "disk1:users1" has been damaged, and no backup is available. The following statement re-creates the original datafile (same size) on disk 2:

```
ALTER DATABASE CREATE DATAFILE 'disk1:users1' AS 'disk2:users1';
```

Note: The old datafile is renamed as the new datafile when an ALTER DATABASE CREATE DATAFILE statement is executed.

This statement enables you to create an empty file that matches the lost file. Oracle looks at information in the control file and the data dictionary to obtain size information. Next, you must perform media recovery on the empty datafile. All archived redo logs written since the original datafile was created must be mounted and reapplied to the new, empty version of the lost datafile during recovery. If the database was created in NOARCHIVELOG mode, the original datafiles of the SYSTEM tablespace cannot be restored using an ALTER DATABASE CREATE DATAFILE statement because the necessary archived redo logs are not available.

Restoring Necessary Archived Redo Log Files

All archived redo log files required for the pending media recovery eventually need to be on disk, so that they are readily available to Oracle.

To determine which archived redo log files you need, you can use the tables V$LOG_HISTORY and V$RECOVERY_LOG. V$LOG_HISTORY lists all of the archived logs, including their probable names, given the current archived log file naming scheme (as set by the parameter LOG_ARCHIVE_FORMAT). V$RECOVERY_LOG lists only the archived redo logs that Oracle needs to perform recovery. It also includes the probable names of the files, using LOG_ARCHIVE_FORMAT. Be aware that you will need all the redo information from the time the datafile was added to the database.

If space is available, restore all of the required archived redo log files to the location currently specified by the initialization parameter LOG_ARCHIVE_DEST. By doing this, you enable Oracle to locate automatically the correct archived redo log file when required during media recovery. If sufficient space is not available at the location indicated by LOG_ARCHIVE_DEST, you can restore some or all of the required archived redo log files to any disk accessible to Oracle. In this case, you can specify the location of the archived redo log files before or during media recovery.

After an archived log is applied, you can delete the restored copy of the archived redo log file to free disk space. However, make sure that a copy of each archived log group still exists on offline storage.

See Also: For more information about tables, see the *Oracle8 Reference*.

Starting Media Recovery

If a damaged database is in ARCHIVELOG mode, it is a candidate for either complete media recovery or incomplete media recovery operations. To begin media recovery operations, use one of the following options of Enterprise Manager:

- the Apply Recovery Archives dialog box
- the Enterprise Manager RECOVER command
- the SQL command ALTER DATABASE

To start any type of media recovery, you must have administrator privileges. All recovery sessions must be compatible. One session cannot start complete media recovery while another performs incomplete media recovery. Also, you cannot start media recovery if you are connected to the database via a multi-threaded server process.

Performing Complete Media Recovery

This section describes the steps necessary to complete media recovery operations, and includes the following topics:

- Performing Closed Database Recovery
- Performing Open-Database, Offline-Tablespace Recovery
- Performing Open-Database, Offline-Tablespace Individual Recovery

Do not depend solely on the steps in the following procedures to understand all the tasks necessary to recover from a media failure. If you haven't already done so, familiarize yourself with the fundamental recovery concepts and strategies in Chapter 4.

Performing Closed Database Recovery

This section describes steps to perform closed database recovery of either all damaged datafiles in one operation, or individual recovery of each damaged datafile in separate operations.

To Perform Closed Database Recovery

1. If the database is open, shut it down using the Enterprise Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. If you're recovering from a media error, correct it if possible.

   Attention: If the hardware problem that caused the media failure was temporary, and the data was undamaged (for example, a disk or controller power failure), *stop at this point*.

3. If files are permanently damaged, restore the most recent backup files (taken as part of a full or partial backup) of *only* the datafiles damaged by the media failure. Do not restore any undamaged datafiles or any online redo log files. If the hardware problem has been repaired, and damaged datafiles can be restored to their original locations, do so, and skip Step 6 of this procedure. If the hardware problem persists, restore the datafiles to an alternative storage device of the database server and continue with this procedure.

   Note: If you do not have a backup of a specific datafile, you might be able to create an empty replacement file that can be recovered.

4. Start Enterprise Manager and connect to Oracle with administrator privileges.
5. Start a new instance and mount, but do not open, the database using either the Enterprise Manager Startup Database dialog box (with the Startup Mount radio button selected), or the STARTUP command with the MOUNT option.
6. If one or more damaged datafiles were restored to alternative locations in Step 3, the new location of these files must be indicated to the control file of the associated database. Therefore, use the operation described in "Renaming and Relocating Datafiles" in the *Oracle8 Administrator's Guide*, as necessary.
7. All datafiles you want to recover must be online during complete media recovery. To get the datafile names, check the list of datafiles that normally accompanies the current control file, or query the V$DATAFILE view. Then, issue the ALTER DATABASE command with the DATAFILE ONLINE option to ensure that all datafiles of the database are online. For example, to guarantee that a datafile named USERS1 (a fully specified filename) is online, enter the following statement:

```
ALTER DATABASE DATAFILE 'users1' ONLINE;
```

If a specified datafile is already online, Oracle ignores the statement.

8. To start closed database recovery of all damaged datafiles in one step, use either the Enterprise Manager Apply Recovery Archive dialog box, or an equivalent RECOVER DATABASE statement.

   To start closed database recovery of an individual damaged datafile, use the RECOVER DATAFILE statement in Enterprise Manager.

   Note: For maximum performance, use parallel recovery to recover the datafiles.

9. Now Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle prompts you for each required redo log file.

Oracle continues until all required archived redo log files have been applied to the restored datafiles. The online redo log files are then automatically applied to the restored datafiles and notifies you when media recovery is complete. If no archived redo log files are required for complete media recovery, Oracle does not prompt for any. Instead, all necessary online redo log files are applied, and media recovery is complete.

After performing closed database recovery, the database is recovered up to the moment that media failure occurred. You can then open the database using the SQL command ALTER DATABASE with the OPEN option.

See Also: For more information about applying redo log files, see "Applying Redo Log Files" on page 12-4.

Performing Open-Database, Offline-Tablespace Individual Recovery

At this point, an open database has experienced a media failure, and the database remains open while the undamaged datafiles remain online and available for use. The damaged datafiles are automatically taken offline by Oracle.

This procedure cannot be used to perform complete media recovery on the datafiles of the SYSTEM tablespace. If the media failure damages any datafiles of the SYSTEM tablespace, Oracle automatically shuts down the database.

See Also: To proceed with complete media recovery, follow the procedure in "Performing Closed Database Recovery" on page 12-15.

To Perform Open-Database, Offline-Tablespace Individual Recovery

1. The starting point for this recovery operation can vary, depending on whether you left the database open after the media failure occurred.
   - *If the database was shut down*, start a new instance, and mount and open the database. Perform this operation using the Enterprise Manager Startup Database dialog box (with the Startup Open radio button selected), or with the STARTUP command with the OPEN option. After the database is open, take all tablespaces that contain damaged datafiles offline.
   - *If the database is still open* and only damaged datafiles of the database are offline, take all tablespaces containing damaged datafiles offline. Oracle identifies damaged datafiles via error messages. Tablespaces can be taken offline using either the Take Offline menu item of Enterprise Manager, or the SQL command ALTER TABLESPACE with the OFFLINE option, as described in "Backing Up Offline Tablespaces and Datafiles" on page 11-8. If possible, take the damaged tablespaces offline with temporary priority (to minimize the amount of recovery).
2. Correct the hardware problem that caused the media failure. If the hardware problem cannot be repaired quickly, you can proceed with database recovery by restoring damaged files to an alternative storage device.
3. If files are permanently damaged, restore the most recent backup files of *only* the datafiles damaged by the media failure. Do not restore undamaged datafiles, online redo log files, or control files. If the hardware problem has been repaired and the datafiles can be restored to their original locations, do so. If the hardware problem persists, restore the datafiles to an alternative storage device of the database server.

   Note: If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

4. If one or more damaged datafiles were restored to alternative locations (Step 3), indicate the new locations of these files to the control file of the associated database by using the procedure in "Renaming and Relocating Datafiles" in the *Oracle8 Administrator's Guide*, as necessary.

5. After connecting with administrator privileges, use the RECOVER TABLESPACE statement in Enterprise Manager to start offline tablespace recovery of all damaged datafiles in one or more offline tablespaces using one step.

   Note: For maximum performance, use parallel recovery to recover the datafiles.

6. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the applying of files is automated, Oracle prompts for each required redo log file.

   Oracle continues until all required archived redo log files have been applied to the restored datafiles. The online redo log files are then automatically applied to the restored datafiles to complete media recovery.

   If no archived redo log files are required for complete media recovery, Oracle does not prompt for any. Instead, all necessary online redo log files are applied, and media recovery is complete.

7. The damaged tablespaces of the open database are now recovered up to the moment that media failure occurred. You can bring the offline tablespaces online using the Place Online menu item of Enterprise Manager, or the SQL command ALTER TABLESPACE with the ONLINE option.

See Also: For more information about redo log application, see "Applying Redo Log Files" on page 12-4.

For more information about creating datafiles, see the *Oracle8 Administrator's Guide*.

Performing Open-Database, Offline-Tablespace Individual Recovery

Identical to the preceding operation, here an open database has experienced a media failure, and remains open while the undamaged datafiles remain online and available for use. The damaged datafiles are automatically taken offline by Oracle.

Note: This procedure cannot be used to perform complete media recovery on the datafiles of the SYSTEM tablespace. If the media failure damages any datafiles of the SYSTEM tablespace, Oracle automatically shuts down the database.

To Perform Open-Database, Offline-Tablespace Individual Recovery

1. The starting point for this recovery operation can vary, depending on whether you left the database open after the media failure occurred.
   - *If the database was shut down*, start a new instance, and mount and open the database. Perform this operation using the Enterprise Manager Startup Database dialog box (with the Startup Open radio button selected), or with the STARTUP command with the OPEN option. After the database is open, take all tablespaces that contain damaged datafiles offline.
   - *If the database is still open* and only damaged datafiles of the database are offline, take all tablespaces containing damaged datafiles offline. Oracle identifies damaged datafiles via error messages. Tablespaces can be taken offline using either the Take Offline menu item of Enterprise Manager, or the SQL command ALTER TABLESPACE with the OFFLINE option. If possible, take the damaged tablespaces offline with temporary priority (to minimize the amount of recovery).
2. Correct the hardware problem that caused the media failure. If the hardware problem cannot be repaired quickly, you can proceed with database recovery by restoring damaged files to an alternative storage device.
3. If files are permanently damaged, restore the most recent backup files (taken as part of a full or partial backup) of *only* the datafiles damaged by the media failure. Do not restore undamaged datafiles, online redo log files, or control files. If the hardware problem has been repaired and the datafiles can be restored to their original locations, do so. If the hardware problem persists, restore the datafiles to an alternative storage device of the database server.

   Note: If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

4. If one or more damaged datafiles were restored to alternative locations (Step 3), indicate the new locations of these files to the control file of the associated database.
5. After connecting with administrator privileges use the RECOVER DATAFILE statement in Enterprise Manager to start recovery of an individual damaged datafile in an offline tablespace.

   Note: For maximum performance, use parallel recovery to recover the datafiles.

6. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle prompts for each required redo log file.

   Oracle continues until all required archived redo log files have been applied to the restored datafiles. The online redo log files are then automatically applied to the restored datafiles to complete media recovery.

   If no archived redo log files are required for complete media recovery, Oracle does not prompt for any. Instead, all necessary online redo log files are applied, and media recovery is complete.

7. The damaged tablespaces of the open database are now recovered up to the moment that media failure occurred. You can bring the offline tablespaces online using the Place Online menu item of Enterprise Manager, or the SQL command ALTER TABLESPACE with the ONLINE option.

See Also: For information about how to proceed with complete media recovery, see "Performing Closed Database Recovery" on page 12-15.

For more information about creating datafiles, see the *Oracle8 Administrator's Guide*.

Performing Incomplete Media Recovery

This section describes the steps necessary to complete the different types of incomplete media recovery operations, and includes the following topics:

- Performing Cancel-based Recovery
- Performing Time-based Recovery
- Performing Change-based Recovery

See Also: Do not rely solely on this section to understand the procedures necessary to recover from a media failure.

Changing the System Time on a Running Database

If your database is affected by seasonal time changes (for example, daylight savings time), you may experience a problem if a time appears twice in the redo log and you want to recover to the second, or later time. To deal with time changes, perform cancel-based or change-based recovery to the point in time where the clock is set back, then continue with the time-based recovery to the exact time.

Performing Cancel-based Recovery

This section describes how to perform cancel-based recovery.

To Perform Cancel-based Recovery

1. If the database is still open and incomplete media recovery is necessary, shut down the database using the Enterprise Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. Make a whole backup of the database (all datafiles, a control file, and the parameter files of the database) as a precautionary measure, in case an error is made during the recovery procedure.
3. If a media failure occurred, correct the hardware problem that caused the media failure.
4. If the current control files do not match the physical structure of the database at the intended time of recovery (for example, if a datafile was added after the point in time to which you intend to recover), then restore a backup of the control file that reflects the database's physical file structure (contains the names of datafiles and online redo log files) at the point at which incomplete media recovery is intended to finish. Review the list of files that correspond to the current control file as well as each control file backup to determine the correct control file to use. If necessary, replace all current control files of the database with the correct control file backup. You can, alternatively, create a new control file to replace the missing one.

   Note: If a database control file cannot function or be replaced with a control file backup, you must edit the parameter file associated with the database to modify the CONTROL_FILES parameter.

5. Restore backup files (taken as part of a full or partial backup) of *all* the datafiles of the database. All backup files used to replace existing datafiles must have been taken before the intended time of recovery. For example, if you intend to recover to redo log sequence number 38, then restore all datafiles with backups completed before redo log sequence number 38.

   If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

If a datafile was added after the intended time of recovery, it is not necessary to restore a backup for this file, as it will no longer be used for the database after recovery is complete.

If the hardware problem that caused a media failure has been solved and all datafiles can be restored to their original locations, do so, and skip Step 8 of this procedure. If a hardware problem persists, restore damaged datafiles to an alternative storage device.

Note: Files in read-only tablespaces should be offline if you are using a control file backup. Otherwise, recovery will try to update the headers of the read-only files.

---

6. Start Enterprise Manager and connect to Oracle with administrator privileges.
7. Start a new instance and mount the database. You can perform this operation using the Enterprise Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.
8. If one or more damaged datafiles were restored to alternative locations in Step 5, the new locations of these files must be indicated to the control file of the associated database.
9. If a backup of the control file is being used with this incomplete recovery (that is, a control file backup or re-created control file was restored in Step 4), indicate this in the dialog box or command used to start recovery (that is, specify the USING BACKUP CONTROLFILE parameter).
10. Use Enterprise Manager Apply Recovery Archives dialog box, or an equivalent RECOVER DATABASE UNTIL CANCEL statement to begin cancel-based recovery.
11. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle supplies the name it expects to find from LOG_ARCHIVE_DEST and requests you to stop or proceed with applying the log file. If the control file is a backup file, you must supply names of online logs.

    Oracle continues to apply redo log files.
12. Continue applying redo log files until the most recent, undamaged redo log file has been applied to the restored datafiles.

13. Enter "CANCEL" to cancel recovery after Oracle has applied the redo log file just prior to the damaged file. Cancel-based recovery is now complete.

    Oracle returns a message indicating whether recovery is successful.

Opening the Database After Successful Cancel-based Recovery

The first time you open the database subsequent to incomplete media recovery, you must explicitly specify whether to reset the log sequence number by including either the RESETLOGS or NORESETLOGS option. Resetting the redo log:

- discards any redo information that was not applied during recovery, ensuring that it will never be applied
- reinitializes the control file information about online redo logs and redo threads
- clears the contents of the online redo logs
- creates the online redo log files if they do not currently exist
- resets the log sequence number to 1

> **WARNING: Resetting the redo log discards all changes to the database made since the first discarded redo information. Updates entered after that time must be re-entered manually.**

Use the following rules when deciding to specify RESETLOGS or NORESETLOGS:

- Reset the log sequence number if you used a backup of the control file in recovery, no matter what type of recovery was performed (complete or incomplete).
- Reset the log sequence number if the recovery was actually incomplete. For example, you must have specified a previous time or SCN, not one in the future.
- Do not reset logs if recovery was complete (unless you used a backup control file). This applies when you intentionally performed complete recovery and when you performed incomplete recovery but actually recovered all changes in the redo logs anyway. See the explanation in step 12 for how to examine the ALERT file to see if incomplete recovery was actually complete.
- Do not reset logs if you are using the archived logs of this database for a standby database. If the log must be reset, then you will have to re-create your standby database.

To preserve the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN NORESETLOGS option. To reset the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN RESETLOGS option. (If you attempt to reset the log when you should not, or if you neglect to reset the log when you should, Oracle returns an error and does not open the database. Correct the error and try again.)

If the log sequence number is reset when opening a database, different messages are returned, depending on whether the recovery was complete or incomplete. If the recovery was complete, the following message appears in the ALERT file:

```
RESETLOGS after complete recovery through change scn
```

If the recovery was incomplete, the following message is reported in the ALERT file:

```
RESETLOGS after incomplete recovery UNTIL CHANGE scn
```

If you reset the redo log sequence when opening the database, immediately shut down the database normally and make a whole database backup. Otherwise, you will not be able to recover changes made after you reset the logs. Until you take a whole database backup, the only way to recover will be to repeat the procedures you just finished, up to resetting the logs. (You do not need to back up the database if you did not reset the log sequence.)

After opening the database using the RESETLOGS option, check the ALERT log to see if Oracle has detected inconsistencies between the data dictionary and the control file (for example, a datafile that the data dictionary includes but does not list in the new control file).

If a datafile exists in the data dictionary but not in the new control file, Oracle creates a placeholder entry in the control file under MISSING*nnnn* (where *nnnn* is the file number in decimal). MISSING*nnnn* is flagged in the control file as being offline and requiring media recovery. The actual datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn*, so that it points to the datafile only when the datafile was read-only or offline normal. If, on the other hand, MISSING*nnnn* corresponds to a datafile that was not read-only or offline normal, then the rename operation cannot be used to make the datafile accessible, because the datafile requires media recovery that is precluded by the results of RESETLOGS. In this case, the tablespace containing the datafile must be dropped.

In contrast, if a datafile indicated in the control file is not present in the data dictionary, Oracle removes references to it from the new control file. In both cases, Oracle includes an explanatory message in the ALERT file to let you know what was found.

See Also: For more information about creating datafiles, see "Restoring Damaged Datafiles" on page 12-13.

To relocate or rename datafiles, see the *Oracle8 Administrator's Guide*.

For more information about applying redo logs, see "Applying Redo Log Files" on page 12-4.

Performing Time-based Recovery

When you are performing time-based, incomplete media recovery, and you are recovering with a backup control file and have read-only tablespaces, contact Oracle Support before attempting this recovery procedure.

To Perform Time-based Recovery

1. If the database is still open and incomplete media recovery is necessary, shut down the database using the Enterprise Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. Make a whole backup of the database (all datafiles, a control file, and the parameter files of the database) as a precautionary measure, in case an error is made during the recovery procedure.
3. If a media failure occurred, correct the hardware problem that caused the media failure.
4. If the current control files do not match the physical structure of the database at the intended time of recovery (for example, if a datafile was added after the point in time to which you intend to recover), then restore a backup of the control file that reflects the database's physical file structure (contains the names of datafiles and online redo log files) at the point at which incomplete media recovery is intended to finish. Review the list of files that corresponds to the current control file and each control file backup to determine the correct control file to use. If necessary, replace all current control files of the database with the

correct control file backup. You can, alternatively, create a new control file to replace the missing one.

> **Note:** If a database control file cannot function or be replaced with a control file backup because the hardware problem causing the media failure persists, you must edit the parameter file associated with the database to modify the CONTROL_FILES parameter.

5. Restore backup files of *all* the datafiles of the database. All backup files used to replace existing datafiles must have been taken before the intended time of recovery. For example, if you intend to recover to redo log sequence number 38, then restore all datafiles with backups completed before redo log sequence number 38.

   If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

   If a datafile was added after the intended time of recovery, it is not necessary to restore a backup for this file, as it will no longer be used for the database after recovery is complete.

   If the hardware problem that caused a media failure has been solved and all datafiles can be restored to their original locations, do so, and skip Step 8 of this procedure. If a hardware problem persists, restore damaged datafiles to an alternative storage device.

   > **Note:** Files in read-only tablespaces should be offline if you are using a control file backup. Otherwise, the recovery will try to update the headers of the read-only files.

6. Start Enterprise Manager and connect to Oracle with administrator privileges.
7. Start a new instance and mount the database. This operation can be performed with the Enterprise Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.
8. If one or more damaged datafiles were restored to alternative locations in Step 5, the new locations of these files must be indicated to the control file of the associated database.

9. All datafiles of the database must be online unless an offline tablespace was taken offline normally. To get the names of all datafiles to recover, check the list of datafiles that normally accompanies the control file being used or query the V$DATAFILE view. Then, use the ALTER DATABASE command and the DATAFILE ONLINE option to make sure that all datafiles of the database are online. For example, to guarantee that a datafile named USERS1 (a fully specified filename) is online, enter the following statement:

   ```
   ALTER DATABASE DATAFILE 'users1' ONLINE;
   ```

   If a backup of the control file is being used with this incomplete recovery (that is, a control file backup or re-created control file was restored), indicate this in the dialog box or command used to start recovery. If a specified datafile is already online, Oracle ignores the statement.

10. Issue the RECOVER DATABASE UNTIL TIME statement to begin time-based recovery. The time is always specified using the following format, delimited by single quotation marks: 'YYYY-MM-DD:HH24:MI:SS'.

11. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle supplies the name it expects to find from LOG_ARCHIVE_DEST and requests you to stop or proceed with applying the log file. If the control file is a backup file, you must supply names of online logs. Oracle continues to apply redo log files.

12. Continue applying redo log files until the last required redo log file has been applied to the restored datafiles. Oracle automatically terminates the recovery when it reaches the correct time, and returns a message indicating whether recovery is successful.

Opening the Database After Successful Time-based Recovery

The first time you open the database subsequent to incomplete media recovery, you must explicitly specify whether to reset the log sequence number by including either the RESETLOGS or NORESETLOGS option. Resetting the redo log:

- discards any redo information that was not applied during recovery, ensuring that it will never be applied
- reinitializes the control file information about online redo logs and redo threads
- clears the contents of the online redo logs
- creates the online redo log files if they do not currently exist

- resets the log sequence number to 1

  WARNING: Resetting the redo log discards all changes to the database made since the first discarded redo information. Updates entered after that time must be re-entered manually.

  Use the following rules when deciding to specify RESETLOGS or NORESETLOGS:

- Reset the log sequence number if you used a backup of the control file in recovery, no matter what type of recovery was performed (complete or incomplete).
- Reset the log sequence number if the recovery was actually incomplete. For example, you must have specified a previous time or SCN, not one in the future.
- Do not reset logs if recovery was complete (unless you used a backup control file). This applies when you intentionally performed complete recovery and when you performed incomplete recovery but actually recovered all changes in the redo logs anyway. See the explanation in step 12 for how to examine the ALERT file to see if incomplete recovery was actually complete.
- Do not reset logs if you are using the archived logs of this database for a standby database. If the log must be reset, then you will have to re-create your standby database.

  To preserve the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN NORESETLOGS option. To reset the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN RESETLOGS option. (If you attempt to reset the log when you should not, or if you neglect to reset the log when you should, Oracle returns an error and does not open the database. Correct the error and try again.)

  If the log sequence number is reset when opening a database, different messages are returned, depending on whether the recovery was complete or incomplete. If the recovery was complete, the following message appears in the ALERT file:

  ```
  RESETLOGS after complete recovery through change scn
  ```

  If the recovery was incomplete, the following message is reported in the ALERT file:

```
RESETLOGS after incomplete recovery UNTIL CHANGE scn
```

If you reset the redo log sequence when opening the database, immediately shut down the database normally and make a whole database backup. Otherwise, you will not be able to recover changes made after you reset the logs. Until you take a whole database backup, the only way to recover will be to repeat the procedures you just finished, up to resetting the logs. (You do not need to back up the database if you did not reset the log sequence.)

After opening the database using the RESETLOGS option, check the ALERT log to see if Oracle has detected inconsistencies between the data dictionary and the control file (for example, a datafile that the data dictionary includes but does not list in the new control file).

If a datafile exists in the data dictionary but not in the new control file, Oracle creates a placeholder entry in the control file under MISSING*nnnn* (where *nnnn* is the file number in decimal). MISSING*nnnn* is flagged in the control file as being offline and requiring media recovery. The actual datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn*, so that it points to the datafile only when the datafile was read-only or offline normal. If, on the other hand, MISSING*nnnn* corresponds to a datafile that was not read-only or offline normal, then the rename operation cannot be used to make the datafile accessible, because the datafile requires media recovery that is precluded by the results of RESETLOGS. In this case, the tablespace containing the datafile must be dropped.

In contrast, if a datafile indicated in the control file is not present in the data dictionary, Oracle removes references to it from the new control file. In both cases, Oracle includes an explanatory message in the ALERT file to let you know what was found.

See Also: For more information see the *Oracle8 Administrator's Guide*.

For more information about applying redo logs, see "Applying Redo Log Files" on page 12-4.

Performing Change-based Recovery

This section describes how to perform change-based recovery.

To Perform Change-based Recovery

1. If the database is still open and incomplete media recovery is necessary, shut down the database using the Enterprise Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. Make a whole backup of the database (all datafiles, a control file, and the parameter files of the database) as a precautionary measure, in case an error is made during the recovery procedure.
3. If a media failure occurred, correct the hardware problem that caused the media failure.
4. If the current control files do not match the physical structure of the database at the intended time of recovery (for example, if a datafile was added after the point in time to which you intend to recover), then restore a backup of the control file that reflects the database's physical file structure (contains the names of datafiles and online redo log files) at the point at which incomplete media recovery is intended to finish. Review the list of files that correspond to the current control file as well as each control file backup to determine the correct control file to use. If necessary, replace all current control files of the database with the correct control file backup. You can, alternatively, create a new control file to replace the missing one.

   Note: If a database control file cannot function or be replaced with a control file backup, you must edit the parameter file associated with the database to modify the CONTROL_FILES parameter.

5. Restore backup files of *all* the datafiles of the database. All backup files used to replace existing datafiles must have been taken before the intended time of recovery. For example, if you intend to recover to redo log sequence number 38, then restore all datafiles with backups completed before redo log sequence number 38.

   If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

If a datafile was added after the intended time of recovery, it is not necessary to restore a backup for this file, as it will no longer be used for the database after recovery is complete.

If the hardware problem that caused a media failure has been solved and all datafiles can be restored to their original locations, do so, and skip Step 8 of this procedure. If a hardware problem persists, restore damaged datafiles to an alternative storage device.

Note: Files in read-only tablespaces should be offline if you are using a control file backup. Otherwise, recovery will try to update the headers of the read-only files.

---

6. Start Enterprise Manager and connect to Oracle with administrator privileges.
7. Start a new instance and mount the database. You can perform this operation using the Enterprise Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.
8. If one or more damaged datafiles were restored to alternative locations in Step 5, the new locations of these files must be indicated to the control file of the associated database.
9. To get the names of all datafiles to recover, check the list of datafiles that normally accompany the control file being used or query the V$DATAFILE view. Then, use the ALTER DATABASE command with the DATAFILE ONLINE option to make sure that all datafiles of the database are online. For example, to guarantee that a datafile named USERS1 (a fully specified filename) is online, enter the following statement:

   ```
   ALTER DATABASE DATAFILE 'users1' ONLINE;
   ```

   If a specified datafile is already online, Oracle ignores the statement.

   If a backup of the control file is being used with this incomplete recovery (that is, a control file backup or re-created control file was restored), specify the USING BACKUP CONTROLFILE parameter in the dialog box or command used to start recovery.
10. Issue the RECOVER DATABASE UNTIL CHANGE statement to begin change-based recovery. The SCN is specified as a decimal number without quotation marks.

11. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle supplies the name it expects to find from LOG_ARCHIVE_DEST and requests you to stop or proceed with applying the log file. If the control file is a backup file, you must supply names of online logs. Oracle continues to apply redo log files.
12. Continue applying redo log files until the last required redo log file has been applied to the restored datafiles. Oracle automatically terminates the recovery when it reaches the correct time, and returns a message indicating whether recovery is successful.

Opening the Database After Successful Change-based Recovery

The first time you open the database subsequent to incomplete media recovery, you must explicitly specify whether to reset the log sequence number by including either the RESETLOGS or NORESETLOGS option. Resetting the redo log:

- discards any redo information that was not applied during recovery, ensuring that it will never be applied
- reinitializes the control file information about online redo logs and redo threads
- clears the contents of the online redo logs
- creates the online redo log files if they do not currently exist
- resets the log sequence number to 1

> **WARNING: Resetting the redo log discards all changes to the database made since the first discarded redo information. Updates entered after that time must be re-entered manually.**

Use the following rules when deciding to specify RESETLOGS or NORESETLOGS:

- Reset the log sequence number if you used a backup of the control file in recovery, no matter what type of recovery was performed (complete or incomplete).
- Reset the log sequence number if the recovery was actually incomplete. For example, you must have specified a previous time or SCN, not one in the future.

- Do not reset logs if recovery was complete (unless you used a backup control file). This applies when you intentionally performed complete recovery and when you performed incomplete recovery but actually recovered all changes in the redo logs anyway. See the explanation in step 12 for how to examine the ALERT file to see if incomplete recovery was actually complete.
- Do not reset logs if you are using the archived logs of this database for a standby database. If the log must be reset, then you will have to re-create your standby database.

  To preserve the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN NORESETLOGS option. To reset the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN RESETLOGS option. (If you attempt to reset the log when you should not, or if you neglect to reset the log when you should, Oracle returns an error and does not open the database. Correct the error and try again.)

  If the log sequence number is reset when opening a database, different messages are returned, depending on whether the recovery was complete or incomplete. If the recovery was complete, the following message appears in the ALERT file:

  ```
  RESETLOGS after complete recovery through change scn
  ```

  If the recovery was incomplete, the following message is reported in the ALERT file:

  ```
  RESETLOGS after incomplete recovery UNTIL CHANGE scn
  ```

If you reset the redo log sequence when opening the database, immediately shut down the database normally and make a full database backup. Otherwise, you will not be able to recover changes made after you reset the logs. Until you take a full backup, the only way to recover will be to repeat the procedures you just finished, up to resetting the logs. (You do not need to back up the database if you did not reset the log sequence.)

After opening the database using the RESETLOGS option, check the ALERT log to see if Oracle has detected inconsistencies between the data dictionary and the control file (for example, a datafile that the data dictionary includes but does not list in the new control file).

If a datafile exists in the data dictionary but not in the new control file, Oracle creates a placeholder entry in the control file under MISSING*nnnn* (where *nnnn* is the file number in decimal). MISSING*nnnn* is flagged in the control file as being

offline and requiring media recovery. The actual datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn*, so that it points to the datafile only when the datafile was read-only or offline normal. If, on the other hand, MISSING*nnnn* corresponds to a datafile that was not read-only or offline normal, then the rename operation cannot be used to make the datafile accessible, because the datafile requires media recovery that is precluded by the results of RESETLOGS. In this case, the tablespace containing the datafile must be dropped.

In contrast, if a datafile indicated in the control file is not present in the data dictionary, Oracle removes references to it from the new control file. In both cases, Oracle includes an explanatory message in the ALERT file to let you know what was found.

See Also: For more information about applying redo logs, see "Applying Redo Log Files" on page 12-4.

Preparing for Disaster Recovery

This section describes how to plan for and implement disaster recovery procedures for your primary database, and includes the following topics:

- Planning and Creating a Standby Database
- Altering the Physical Structure of the Primary Database

Planning and Creating a Standby Database

A *standby database* maintains a duplicate, or standby copy of your primary (also known as *production*) database and provides continued primary database availability in the event of a disaster (when all media is destroyed at your production site). A standby database is constantly in recovery mode. If a disaster occurs, you can take the standby database out of recovery mode and activate it for online use. A standby database is intended *only* for recovery of the primary database; you cannot query or open it for any purpose other than to activate disaster recovery. Once you activate your standby database, you cannot return it to standby recovery mode unless you re-create it as another standby database.

WARNING: Activating a standby database resets the online logs of the standby database. Hence, after activation, the logs from your standby database and production database are incompatible.

You must place the data files, log files, and control files of your primary and standby databases on separate physical media. Therefore, it is impossible to use the same control file for both your primary and standby databases.

Creating a Standby Database

This section lists the steps and rules to follow when creating a standby database.

To Create a Standby Database

1. Back up (either online or offline) the data files from your primary database.
2. Create the control file for your standby database by issuing the ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'filename' command, which creates a modified copy of the primary database's control file.
3. Archive the current online logs of the primary database by issuing the ALTER SYSTEM ARCHIVE LOG CURRENT command. Issuing the ALTER SYSTEM ARCHIVE LOG CURRENT command also ensures consistency among the data files in step 1, the control file in step 2, and the log files.
4. Transfer the standby database control file, archived log files, and backed up data files to the remote (standby) site using operating system commands or utilities. Use an appropriate method if transferring binary files.

WARNING: Oracle encourages you to use a datafile naming scheme that keeps the datafile names the same at both the primary and standby databases. If this is not possible, then you can use the datafile name conversion parameters. If you do not use either of these suggested datafile naming schemes, you may end up crashing your standby database.

See Also: For information about setting name conversion parameters when you create your standby database, see "Converting Data File and Log File Names" on page 12-37.

Maintaining a Standby Database

This section provides the tasks for maintaining your standby database, including information about clearing standby logfiles.

To Maintain Your Standby Database In Recovery Mode

1. Start up the Oracle instance at the standby database using the NO MOUNT clause.
2. Issue the ALTER DATABASE MOUNT STANDBY DATABASE [EXCLUSIVE / PARALLEL] command.
3. Transfer the archived redo logs from the primary database to the remote (standby) site. Use an appropriate operating system utility for transferring binary data.
4. Place the standby database in recovery mode by issuing the RECOVER [FROM 'location'] STANDBY DATABASE command.

> **Note:** As the archived logs are generated, you must continually transfer and apply them to the standby database. Also, you can only apply logs that have been archived at the primary database to the standby database.

Clearing Online Logfiles You can clear standby database online logfiles to optimize performance as you maintain your standby database. If you prefer not to perform this operation during maintenance, the online logfiles will be cleared automatically during activation. You can clear logfiles using the following statement:

```
ALTER DATABASE CLEAR LOGFILE GROUP integer;
```

Converting Data File and Log File Names

You can set the following initialization parameters so that all filenames from your primary database control file are converted for use by your standby database:

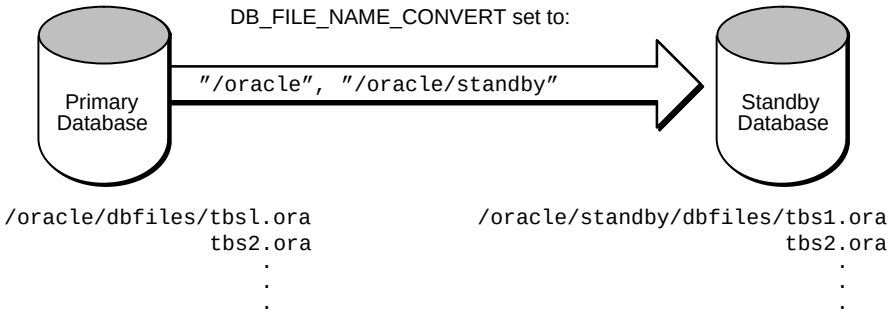
- DB_FILE_NAME_CONVERT
- LOG_FILE_NAME_CONVERT

If your primary and standby databases exist on the same machine (of course, they should not, but if they are), setting these parameters is advisable, because they allow you to make your standby database filenames distinguishable from your primary database filenames.

The DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT parameters must have two strings. The first string is a sequence of characters to be looked for in a primary database filename. If that sequence of characters is matched, it is replaced by the second string to construct the standby database filename.

Figure 12–1 shows how the filename conversion parameters work.

Figure 12–1 Setting Filename Conversion Parameters



Note: If you perform a data file (or log file) RENAME at the standby database, or use the AS clause with the ALTER DATABASE CREATE FILE command, then the conversion parameters will not apply to that file.

Activating a Standby Database

In the event of a disaster, you should (if possible) archive your primary database logs (ALTER SYSTEM ARCHIVE LOG CURRENT), transfer them to your standby site, and apply them before activating your standby database. This makes your standby database current to the same point in time as your primary database (before the failure). If you cannot archive your current online logs, then you must activate the standby database without recovering the transactions from the unarchived logs of the primary database.

After you activate your standby database, its online redo logs are reset. Note that this makes the logs from the standby database and primary database incompatible. Also, the standby database is dismounted when activated, therefore, you are unable to look at tables and views immediately after activation.

1. Ensure that your standby database is mounted in EXCLUSIVE mode.
2. Issue the ALTER DATABASE ACTIVATE STANDBY DATABASE command.
3. Shut down your standby instances.

4. As soon as possible, back up your new production database. At this point, the former standby database is now your production database. This task, while not required, is a recommended safety measure, because you cannot recover changes made after activation without a backup.
5. Startup the new production instance.

> **Note:** After you activate your standby database, all transactions from unarchived logs at your original production database are lost.

Altering the Physical Structure of the Primary Database

Altering the physical structure of your primary database can have an impact on your standby database. The following sections describe the effects of primary database structural alterations on a standby database.

Adding Data Files

Adding a data file to your primary database generates redo information that, when applied at your standby database, automatically adds the data file name to the standby control file. If the standby database locates the new file with the new filename, the recovery process continues. If the standby database is unable to locate the new data file, the recovery process will stop.

If the recovery process stops, then perform *either* of the following procedures before resuming the standby database recovery process:

- Copy a backup of the added data file from the primary database to the standby database.
- Issue the ALTER DATABASE CREATE DATAFILE command at the standby database.

If you don't want the new data file in the standby database, you can take it offline using the DROP option.

See Also: For more information on offline datafile alterations, see "Taking Datafiles in the Standby Database Offline" on page 12-41.

Renaming Files

Data file renames on your primary database do not take effect at the standby database until the standby database control file is refreshed. If you want the data files at your primary and standby databases to remain in sync when you rename

primary database data files, then perform analogous operations on the standby database.

Altering Log Files

You can add log file groups or members to the primary database without affecting your standby database. Likewise, you can drop log file groups or members from the primary database without affecting your standby database. Similarly, enabling and disabling of threads at the primary database has no effect on the standby database.

You may want to keep the online log file configuration the same at the primary and standby databases. If so, when you enable a log file thread with the ALTER DATABASE ENABLE THREAD at the primary database, you should create a new control file for your standby database before activating it. See “Refreshing the Standby Database Control File” on page -42 for refresh procedures.

If you clear log files at the primary database by issuing the ALTER DATABASE CLEAR UNARCHIVED LOGFILE command, or open the primary database using the RESETLOGS option, you invalidate the standby database. Because the standby database recovery process will not have the archived logs it requires to continue, you will need to re-create the standby database.

Altering Control Files

If you use the CREATE CONTROLFILE command at the primary database to perform any of the following, you may invalidate the standby database’s control file:

- change the maximum number of redo log file groups or members
- change the maximum number of instances that can concurrently mount and open the database

If you’ve invalidated the standby database’s control file, you must re-create it using the procedures in “Refreshing the Standby Database Control File” on page 12-42.

Using the CREATE CONTROLFILE command with the RESETLOGS option on your primary database will force the next open of the primary database to reset the online logs, thereby invalidating the standby database.

Configuring Initialization Parameters

Most initialization parameters at your primary and standby databases should be identical. Specific initialization parameters such as CONTROL_FILES and DB_FILE_STANDBY_NAME_CONVERT should be changed. Differences in other

initialization parameters may cause performance degradation at the standby database, and in some cases, bring standby database operations to a halt.

The following initialization parameters play a key role in the standby database recovery process:

- COMPATIBLE

  The COMPATIBLE parameter must be the same at the primary and standby databases. If it is not, you may not be able to apply the logs from your primary database to your standby database.

- DB_FILES

  MAXDATAFILES must be the same at both databases so that you allow the same number of files at the standby database as you allow at the primary database.

- CONTROL_FILES

  CONTROL_FILES must be different between the primary and standby databases. The names of the control files that you list in this parameter for the standby database must exist at the standby database.

- DB_FILE_STANDBY_NAME_CONVERT (or LOG_FILE_STANDBY_NAME_CONVERT)

  Set the DB_FILE_STANDBY_NAME_CONVERT (or LOG_FILE_STANDBY_NAME_CONVERT) parameter when you want to make your standby database filenames distinguishable from your primary database filenames. For more information on this parameter see "Converting Data File and Log File Names" on page 12-37.

See Also: For more information on initialization parameters, see the *Oracle8 Reference*.

Taking Datafiles in the Standby Database Offline

You can take standby database datafiles offline as a means to support a subset of your primary database's datafiles. For example, you decide it is undesirable to recover the primary database's temporary tablespaces on the standby database. So you take the datafiles offline using the ALTER DATABASE DATAFILE 'fn' OFFLINE DROP command on the standby database. If you do this, then the tablespace containing the offline files must be dropped after opening the standby database.

Performing Direct Path Operations

When you perform a direct load originating from either direct path load, table create via subquery, or index create on the primary database, the performance improvement applies *only* to the primary database; there is no corresponding recovery process performance improvement on the standby database. The standby database recovery process still sequentially reads and applies the redo information generated by the unrecoverable direct load.

Primary database processes using the UNRECOVERABLE option are not propagated to the standby database. Why? Because these processes do not appear in the archived redo logs. If you want to propagate such processes to your standby database, perform any *one* of the following tasks.

To Propagate UNRECOVERABLE Processes to Standby Database

1. Take the affected datafiles offline in the standby database, and drop the tablespace after activation.
2. Re-create the standby database from a new database backup.
3. Back up the affected tablespace and archive the current logs in the primary database. Transfer the datafiles to the standby database. Then resume standby recovery. This is the same procedure that you would perform to guarantee ordinary database recoverability after an UNRECOVERABLE operation.

If you perform an unrecoverable operation at the primary database, and attempt to recover at the standby database, you will not receive error messages during recovery. Such error messages appear in the standby database alert log. Thus, you should check the standby database alert log periodically.

See Also: For more details, see “Taking Datafiles in the Standby Database Offline” on page 12-41.

Refreshing the Standby Database Control File

The following steps describe how to refresh, or create a copy of changes you’ve made to the primary database control file.

To Refresh the Standby Database Control File

1. Issue the CANCEL command on the standby database to halt its recovery process.
2. Shut down the standby instances.
3. Issue the ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'filename' statement on the primary database to create the control file for the standby database.
4. Issue the ALTER SYSTEM ARCHIVE LOG CURRENT statement on the primary database to archive the current online logs of your primary database.
5. Transfer the standby control file and archived log files to the standby site.
6. Restart and mount (but do not open) the standby database by issuing the ALTER DATABASE MOUNT STANDBY DATABASE [EXCLUSIVE / PARALLEL] statement.
7. Restart the recovery process on the standby database by issuing the RECOVER [FROM 'location'] STANDBY DATABASE statement.

Unrecoverable Objects and Recovery

You can create tables and indexes using the CREATE TABLE AS SELECT command. You can also specify that Oracle create them as *unrecoverable*. When you create a table or index as unrecoverable, Oracle does not generate redo log records for the operation. Thus, objects created unrecoverable cannot be recovered, even if you are running in ARCHIVELOG mode.

> **Note:** If you cannot afford to lose tables or indexes created unrecoverable, take a backup after the unrecoverable table or index is created.

Be aware that when you perform a media recovery, and some tables or indexes are created as recoverable while others are unrecoverable, the unrecoverable objects will be marked logically corrupt by the RECOVER operation. Any attempt to access the unrecoverable objects returns an ORA-01578 error message. You should drop the unrecoverable objects, and recreate them, if needed.

Because it is possible to create a table unrecoverable and then create a recoverable index on that table, the index is not marked as logically corrupt after you perform a media recovery. However, the table was unrecoverable (and thus marked as

corrupt after recovery), so the index points to corrupt blocks. The index must be dropped, and the table and index must be re-created if necessary.

See Also: You can find more information about the impact of UNRECOVERABLE operations on a standby database on page 12-42.

Read-only Tablespaces and Recovery

This section describes how read-only tablespaces affect instance and media recovery.

Using a Backup Control File

Media recovery with the USING BACKUP CONTROLFILE option checks for read-only files. It is an error to attempt recovery of a read-only file. You can avoid this error by taking all datafiles from read-only tablespaces offline before doing recovery with a backup control file. Therefore, it is very important to have the correct version of the control file for the recovery. If the tablespace will be read-only when the recovery is complete, then the control file must be from a time when the tablespace was read-only. Similarly, if the tablespace will be read-write at the end of recovery, it should be read-write in the control file. If the appropriate control file is not available, you should create a new control file with the CREATE CONTROLFILE command.

Re-creating a Control File

If you need to re-create a control file for a database with read-only tablespaces, you must follow some special procedures. Issue the ALTER DATABASE BACKUP CONTROLFILE TO TRACE command to get a listing of the procedure that you need to follow. The procedure is similar to the procedure for offline normal tablespaces, except that you need to bring the tablespace online after the database is open.

Re-creating a control file can also affect the recovery of read-write tablespaces that were at one time read-only. If you re-create the control file after making the tablespace writable, Oracle can no longer determine when the tablespace was changed from read-only to read-write. Thus, you can no longer recover from the read-only version of the tablespace. Instead, you must recover from the time of the most recent backup. It is important to backup a tablespace immediately after making it read-write.

Recovery Procedure Examples

This section describes how to recover from common media failures, and includes the following topics:

- Types of Media Failures
- Loss of Datafiles
- Loss of Online Redo Log Files
- Loss of Archived Redo Log Files
- Loss of Control Files
- Recovery From User Errors

Types of Media Failures

Media failures fall into two general categories: permanent and temporary. Permanent media failures are serious hardware problems that cause the permanent loss of data on the disk. Lost data cannot be recovered except by repairing or replacing the failed storage device and restoring backups of the files stored on the damaged storage device. Temporary media failures are hardware problems that make data temporarily inaccessible; they do not corrupt the data. Following are two examples of temporary media failures:

- A disk controller fails. Once the disk controller is replaced, the data on the disk can be accessed.
- Power to a storage device is cut off. Once the power is returned, the storage device and all associated data is accessible again.

Loss of Datafiles

If a media failure affects datafiles of a database, the appropriate recovery procedure depends on the archiving mode of the database, the type of media failure, and the exact files affected by the media failure. The following sections explain the appropriate recovery strategies in various situations.

Loss of Datafiles, NOARCHIVELOG Mode

If either a permanent or temporary media failure affects *any* datafiles of a database operating in NOARCHIVELOG mode, Oracle automatically shuts down the database. Depending on the type of media failure, you can use one of two recovery paths:

- If the media failure is temporary, correct the temporary hardware problem and restart the database. Usually, instance recovery is possible, and all committed transactions can be recovered using the online redo log.
- If the media failure is permanent, follow the steps on page -16 to recover from the media failure.

Loss of Datafiles, ARCHIVELOG Mode

If either a permanent or temporary media failure affects the datafiles of a database operating in ARCHIVELOG mode, the following situations can exist:

- If a temporary or permanent media failure affects any datafiles of the SYSTEM tablespace or any datafiles that contain active rollback segments, the database becomes inoperable and should be immediately shut down if it has not already been shut down by Oracle.

  If the hardware problem is temporary, correct the problem and restart the database. Usually, instance recovery is possible, and all committed transactions can be recovered using the online redo log.

  If the hardware problem is permanent, follow the procedure in "Performing Closed Database Recovery" on page 12-15.

- If a temporary or permanent media failure affects only datafiles not mentioned in the previous item, the affected datafiles are unavailable and taken offline automatically by Oracle, but the database can continue to operate.

  If the unaffected portions of the database must remain available, do not shut down the database. First take all tablespaces that contain problem datafiles offline using the temporary option. Then follow the procedure in "Performing Closed Database Recovery" on page 12-15.

Loss of Online Redo Log Files

If a media failure has affected the online redo log of a database, the appropriate recovery procedure depends on the configuration of the online redo log (mirrored or non-mirrored), the type of media failure (temporary or permanent), and the types of online redo log files affected by the media failure (current, active, not yet archived, or inactive online redo log files). The following sections describe the appropriate recovery strategies in various situations.

Loss of an Online Redo Log Member of Mirrored Online Redo Log

If the online redo log of a database is mirrored, and at least one member of each online redo log group is not affected by the media failure, Oracle allows the database to continue functioning as normal (error messages are written to the LGWR trace file and ALERT file of the database). However, you should handle the problem by taking one of the following actions:

- If the hardware problem is temporary, correct the problem. After it has been fixed, LGWR accesses the previously unavailable online redo log files as if the problem never existed.
- If the hardware problem is permanent, use the DROP command to drop the damaged member and use the ADD command to add a new member.

> **Note:** The newly added member provides no redundancy until the log group is reused.

Loss of All Online Redo Log Members of an Online Redo Log Group

If all members of an online redo log group are damaged by a media failure, different situations can arise, depending on the type of online redo log group affected by the failure and the archiving mode of the database. You can locate the filename in V$LOGFILE, and then look for the group number corresponding to the one you lost to verify the lost file's status (verify that it was inactive).

```
SELECT *
FROM v$logfile
;

GROUP#    STATUS       MEMBER
-------   -----------  ---------------------
0001                    log1
0002                    log2
0003                    log3

SELECT *
FROM v$log
;

GROUP#  MEMBERS          STATUS     ARCHIVED
------  -------          ---------  -----------
 0001   1                INACTIVE   YES
 0002   1                ACTIVE     YES
 0003   1                CURRENT    NO
```

Loss of an Inactive, Online Redo Log Group If all members of an inactive online redo log group are damaged, the following situations can arise:

- If a temporary media failure affects only an inactive online redo log group, correct the problem; LGWR can reuse the group when required.
- If a media failure permanently prevents access to only an inactive online redo log group, the damaged inactive online redo log group will eventually halt normal database operation.

  If you notice the problem before the database shuts down, use the ALTER DATABASE CLEAR LOGFILE command.

  If the database has already shut down, perform the following tasks:

To Recover From Loss of an Inactive, Online Redo Log Group

1. Abort the current instance immediately with the Enterprise Manager Shutdown Database dialog box with the Shutdown Abort radio button selected, or the SHUTDOWN command with the ABORT option.
2. Start a new instance and mount the database, but do not open it. This operation can be performed with the Enterprise Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.
3. If the lost log was archived, issue the ALTER DATABASE CLEAR LOGFILE command.
4. If the lost log was unarchived, issue the ALTER DATABASE CLEAR UNARCHIVED LOGFILE command, and immediately backup the database. Also backup the database's control file (using the ALTER DATABASE command with the BACKUP CONTROLFILE option).

   Clearing a log that has not been archived allows it to be reused without archiving it. However, this will make backups unusable if they were started before the last change in the log (unless the file was taken offline prior to the first change in the log). Hence, if the cleared logfile is needed for recovery of a backup, it will not be possible to recover that backup.

   If there is an offline datafile that requires the cleared unarchived log to bring it online, the keywords UNRECOVERABLE DATAFILE are required. The datafile

and its entire tablespace will have to be dropped from the database because the redo necessary to bring it online is being cleared, and there is no copy of it.

Note: The ALTER DATABASE CLEAR LOGFILE command could fail (with an I/O error due to media failure) in two cases:

- When it is not possible to relocate the logfile onto alternative media by re-creating it under the currently configured logfile name.
- When it is not possible to reuse the currently configured logfile name to recreate the logfile because the name itself is invalid or unusable (for example, due to media failure).

In these two cases, the CLEAR LOGFILE command (before receiving the I/O error) would have successfully updated the control file to change the state of the logfile to "being cleared" and "not requiring archiving." The I/O error occurred at the step in which CLEAR LOGFILE attempts to create the new logfile and write zeros to it.

At this point, you can complete recovery by executing, in order, the following commands:

- ADD a logfile under a new name.
- DROP the logfile under the old name.

You can now open the database.

Loss of an Active Online Redo Log Group If your database is still running and the lost active log is not the current log, you can use the ALTER SYSTEM CHECKPOINT command. If successful, your active log is rendered inactive, and you can follow the steps on page 12-48.

If unsuccessful, or if your database has already halted, you cannot use the steps on page 12-48. Instead, perform the following tasks:

To Recover From Loss of an Active Online Redo Log Group

1. If the media failure is temporary, correct the problem so that Oracle can reuse the group when required.
2. If the database is in NOARCHIVELOG mode and a permanent media failure prevents access to an active online redo log group, restore the database from a whole database backup.

   After restoring the database, redo the work and open the database using the RESETLOGS option. Updates done after the backup have been lost and must be re-executed. Shut down the database and take a whole database backup.
3. If the database was in ARCHIVELOG mode, incomplete media recovery must be performed. Use the procedure given in "Performing Complete Media Recovery" on page 12-15, recovering up through the log before the damaged log. Ensure that the current name of the lost redo log can be used for a newly created file. If not, issue the RENAME command to rename the damaged online redo log group to a new location.
4. Open the database using the RESETLOGS option.

   Note: All updates executed from the endpoint of the incomplete recovery to the present must be re-executed.

Loss of Multiple Redo Log Groups If you have lost multiple groups of the online redo log, use the recovery method for the most difficult log to recover. The order of difficulty, from most difficult to least, follows:

1. the current online redo log
2. the active online redo log
3. the unarchived redo log
4. the inactive online redo log

Loss of Archived Redo Log Files

If the database is operating so that filled online redo log groups are being archived, and the only copy of an archived redo log file is damaged, it does not affect the present operation of the database. However, the following situations can arise if media recovery is required in the future:

- If *all* datafiles have been backed up after the filled online redo log group (which is now archived) was written, the archived version of the filled online redo log group is not required for complete media recovery operation.
- Assume the most recent backup file of a datafile was taken before the filled online redo log group was written. The group now corresponds to the damaged archived redo log file. At some future point, the corresponding datafile is damaged by a permanent media failure. The most recent backup of the damaged datafile must be used, and incomplete media recovery can only recover the database up to the damaged archived redo log file.
- If time-based recovery is needed, the damaged archived redo log file may be required if you use old datafile backups that were taken before the original online redo log group was written. In this case, the incomplete media recovery can only recover the database up to the damaged archived redo log group.

> **WARNING: If you know that an archived redo log group has been damaged, immediately backup all datafiles so that you will have a whole database backup that does not require the damaged archived redo log.**

Loss of Control Files

If a media failure has affected the control files of a database (whether control files are mirrored or not), the database continues to run until the first time that an Oracle background process needs to access the control files. At this point, the database and instance are automatically shut down.

If the media failure is temporary and the database has not yet shut down, immediately correcting the media failure can avoid the automatic shut down of the database. However, if the database shuts down before the temporary media failure is corrected, you can restart the database after fixing the problem (and restoring access to the control files).

The appropriate recovery procedure for media failures that permanently prevent access to control files of a database depends on whether you have mirrored the control files. The following sections describe the appropriate procedures.

Loss of a Member of a Mirrored Control File

Use the following steps to recover a database after one or more control files of a database have been damaged by a permanent media failure, and at least one control file has not been damaged by the media failure.

> **Note:** If all control files of a mirrored control file configuration have been damaged, follow the instructions for recovering from the loss of non-mirrored control files.

To Recover a Database After Control Files Are Damaged

1. If the instance is still running, immediately abort the current instance with the Enterprise Manager Shutdown Abort option of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. Correct the hardware problem that caused the media failure. If the hardware problem cannot be repaired quickly, you can proceed with database recovery by restoring damaged control files to an alternative storage device.
3. Use an intact copy of the database's control file to copy over the damaged control files. If possible, copy the intact control file to the original locations of all damaged control files. If the hardware problem persists, copy the intact control file to alternative locations. If you restored *all* damaged control files to their original location, proceed to Step 5. If all damaged control files were not restored, or not restored to their original location, proceed to Step 4.
4. If all damaged control files were not restored, or not restored to their original location in Step 3, the parameter file of the database must be edited so that the CONTROL_FILES parameter reflects the current locations of all control files and excludes all control files that were not restored.
5. Start a new instance. Mount and open the database.

Loss of All Copies of the Current Control File

If all control files of a database have been lost or damaged by a permanent media failure, but all online redo logfiles remain intact, you can recover by creating a new control file (using the CREATE CONTROLFILE command with the NORESETLOGS option). Then execute RECOVER DATABASE followed by ALTER DATABASE OPEN.

Depending on the existence and currency of a control file backup, you have the following options for generating the text of the CREATE CONTROLFILE command:

- If you have executed ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS since you made the last structural change to the database, and if you have saved the SQL command output, then you can use the CREATE CONTROLFILE command from the output as-is. If, however, your most recent execution of ALTER DATABASE BACKUP CONTROLFILE TO TRACE was performed before you made a structural change to the database, then you must edit the output of ALTER DATABASE BACKUP CONTROLFILE TO TRACE to reflect that structural change. For example, if you recently added a datafile to the database, then you should add that datafile to the DATAFILE clause of the CREATE CONTROLFILE command.
- If you have not backed up the control file using the TO TRACE option, but instead have used the TO *filename* option of the ALTER DATABASE BACKUP CONTROLFILE command, then you can use the control file copy to obtain SQL command output. You can do this by copying the backup control file and executing STARTUP MOUNT before executing ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS. If your control file copy predated a recent structural change, you must edit the TO TRACE output to reflect that structural change.
- If you do not have a backup of the control file (in either TO TRACE format or TO *filename* format), then you must generate the CREATE CONTROLFILE command manually.

Recovery From User Errors

An accidental or erroneous operational or programmatic change to the database can cause loss or corruption of data. Recovery may require a return to a state prior to the error.

> **Note:** If the database administrator has properly granted powerful privileges (such as DROP ANY TABLE) to only selected, appropriate users, user errors that require database recovery are minimized.

1. Back up the existing, intact database.
2. Leave the existing database intact, but reconstruct a temporary copy of the database up to the time of the user error using time-based recovery.
3. Export the lost or corrupted data from the reconstructed, temporary copy of the database.

4. Import the lost or corrupted data into the permanent database.
5. Delete the files associated with the temporary copy of the reconstructed database to conserve disk space.

The following scenario describes how to recover a table that has been accidentally dropped.

1. The database that experienced the user error can remain online and available for normal use. The database can remain open or be shut down. Back up all datafiles of the existing database in case an error is made during the remaining steps of this procedure.
2. Create a temporary copy of the database to a past point-in-time using time-based recovery. Be careful not to cause a conflict with the existing control file of the permanent database. Restore a single control file backup to an alternative location (Step 4) and edit the parameter file, as necessary, or create a new control file at the alternative location. Also, restore all datafiles to alternative locations (Step 5) so that you do not affect the permanent copy of the database.
3. Export the lost data using the Oracle utility Export from the temporary, restored version of the database. In this case, export the accidentally dropped table.

   Note: System audit options are exported.

4. Import the exported data (Step 3) into the permanent copy of the database using the Oracle Import utility.
5. Delete the files of the temporary, reconstructed copy of the database to conserve space.

See Also: For more information about the Import and Export utilities, see *Oracle8 Utilities*.

13

Performing Tablespace Point-in-Time Recovery

Attention: Due to the complex nature of tablespace point-in-time recovery, Oracle recommends that you contact and work with Worldwide Customer Support Services before using the procedures described here.

This chapter describes how to perform tablespace point-in-time recovery (TSPITR), and includes the following topics:

- Introduction to Tablespace Point-in-Time Recovery
- Planning for Tablespace Point-in-Time Recovery
- Performing Tablespace Point-In-Time Recovery
- Performing Partial TSPITR of Partitioned Tables
- Performing TSPITR of Partitioned Tables When a Partition Has Been Dropped
- Performing TSPITR of Partitioned Tables When a Partition Has Been Split
- TSPITR Tuning Considerations

Introduction to Tablespace Point-in-Time Recovery

Tablespace Point-In-Time Recovery (TSPITR) enables you to quickly recover one or more tablespaces to a point-in-time that is different from that of the rest of the database. TSPITR is most useful in the following situations:

- To recover from an erroneous drop or truncate table operation.
- To recover a table that has become logically corrupted.
- To recover from an incorrect batch job or other DML statement that has affected only a subset of the database.
- In cases where there are multiple logical databases in separate tablespaces of one physical database, and where one logical database must be recovered to a point different from that of the rest of the physical database.
- For VLDBs (very large databases) even if a full database point-in-time recovery would suffice, you might choose to do tablespace point-in-time recovery rather than restore the whole database from a backup and perform a complete database roll-forward (see "Planning for Tablespace Point-in-Time Recovery" on page 13-3 before making any decisions).

Similar to a table export, TSPITR enables you to recover a consistent data set; however, the data set is the entire tablespace rather than just one object.

Prior to Oracle8, point-in-time recovery could only be used on a subset of a database by:

1. Creating a copy of the database
2. Rolling the copied database forward to the desired point in time
3. Exporting the desired objects from the copied database
4. Dropping the relevant objects from the production database
5. Importing the objects into the production database

However, there was a performance overhead associated with exporting and importing large objects.

TSPITR enables you to recover a subset of a database, and optimizes the export / import phase by enabling you to make an operating system-level datafile copy (of the relevant files of the recovered database) to the production database. Data dictionary information about the file's content (for example, the recovered segments within the file) is transferred to the production database by means of a meta-data export / import from the copied database. The copied file is also added to the production database via this import.

Familiarize yourself with the following terms and abbreviations, which are used throughout this chapter:

TSPITR
Tablespace Point-in-Time Recovery

Clone Database
The copied database used for recovery in Oracle 8 TSPITR is called a "clone database", and has various substantive differences from a regular database.

Recovery Set
Tablespaces that require point-in-time recovery to be performed on them.

Auxiliary Set
Any other items required for TSPITR, including:

- backup control file
- system tablespaces
- datafiles containing rollback segments
- temporary tablespace (optional)

A small amount of space is required by export for sort operations. If a copy of the temporary tablespace is not included in the auxiliary set, then you must provide sort space either by creating a new temporary tablespace after the clone has been started up, or by setting autoextend to ON on the system tablespace files.

Planning for Tablespace Point-in-Time Recovery

TSPITR is a complicated procedure and requires careful planning. Before proceeding you should read all of this chapter thoroughly.

WARNING: You should not perform TSPITR for the first time on a production system, or during circumstances where there is a time constraint.

Limitations Advisory

This section describes the limitations of TSPITR.

The primary issue you should consider when deciding whether or not to perform TSPITR is the possibility of application-level inconsistencies between tables in recovered and unrecovered tablespaces (due to implicit rather than explicit referential dependencies). You must understand these dependencies, and also have the means to resolve any possible inconsistencies before proceeding.

TS_PITR_CHECK Does Not Check for Objects Owned by SYS

The TS_PITR_CHECK view provides information on dependencies and restrictions that can prevent TSPITR from proceeding. However, TS_PITR_CHECK does not provide information about dependencies and restrictions for objects owned by SYS.

If there are any objects, including undo segments, owned by SYS in the recovery set, there is no guarantee that you can successfully recover these objects (because TSPITR utilizes the Export and Import utilities, which do not operate on objects owned by SYS). To find out which recovery set objects are owned by SYS, issue the following statement:

```
SELECT OBJECT_NAME, OBJECT_TYPE
   FROM SYS.DBA_OBJECTS
   WHERE TABLESPACE_NAME IN ('<tablespacename1>','<tablespacename','
   <tablespace name  N')   and owner = 'SYS';
```

See Also: For more details about the TS_PITR_CHECK view, see "Step 2: Research and Resolve Dependencies on the Primary Database" on page 13-8.

TS_PITR_CHECK Does Not Detect Snapshot Tables

The TS_PITR_CHECK view does not detect snapshot tables (it does detect snapshot logs); they are exported as stand-alone tables. Thus, if a snapshot is dropped at time 3, and a backup from time 1 is used to roll forward to time 2, after TSPITR is complete the snapshot table will have been created as a stand-alone table, but without its associated snapshot view.

Partitioned Tables and TS_PITR_CHECK

If any of the tablespaces supplied to the predicate contain the first segment of a partitioned table, then the result set of the TS_PITR_CHECK view is inverted. If tablespaces supplied to the predicate do not include the first segment of a partitioned table, then one row is returned for the partition in question. If the tablespaces supplied to the predicate contain the first segment of a partitioned table, the results are inverted (for example, one row is returned for every tablespace

containing partitions of that partitioned table, but not the tablespace that was supplied to the predicate). Returned rows indicate that there is a conflict that you must resolve by exchanging the partitions with stand-alone tables.

See Also: For more information see "Performing Partial TSPITR of Partitioned Tables" on page 13-16.

Bitmap Indexes

You must drop and re-create bitmap indexes after you complete TSPITR. If you don't, they will be unusable. If any bitmap indexes exist on the tables, imports will fail even if the bitmap indexes have been dropped from the primary database. An incorrect index segment will also be created, despite the failure, and you will have to drop and re-build the index.

Non-Partitioned Global Indexes

The TS_PITR_CHECK view does not detect non-partitioned global indexes of partitioned tables that are outside the recovery set. This is apparent when the view is queried manually and also during the export and import phase of TSPITR. After TSPITR completes, the old index still exists on the recovered table, even though no errors are returned. You must drop and re-create the index.

> **Note:** Because the index is still valid, queries that use the index will return incorrect rows.

General Restrictions

In addition to the preceding limitations, TSPITR has the following restrictions:

- You cannot use TSPITR to recover dropped tablespaces.
- You cannot use TSPITR to recover a tablespace that has been dropped and re-created with the same name.
- You cannot use TSPITR to remove a datafile that has been added to the wrong tablespace. If the file was added after the point to which TSPITR is being performed, then the file will still be part of the tablespace (and will be empty) after TSPITR is complete.
- You cannot use DML statements on the clone database—the clone database is for recovery purposes only.
- After TSPITR is complete, you cannot use existing backups of the recovery set datafiles for recovery; instead, you must take fresh backups of the recovered

files. If you attempt to recover using a backup taken prior to performing TSPITR, recovery will fail (with the error message ORA 1247, 00000, "database recovery through TSPITR of tablespace %s").

- TSPITR does not recover optimizer statistics for objects that have had statistics calculated on them; statistics must be re-calculated after performing TSPITR.
- The following object types are not allowed within the TSPITR recovery set:
  - replicated master tables
  - tables with varray columns
  - tables with nested tables
  - tables with external Bfiles
  - snapshot logs
  - snapshot tables
  - objects owned by SYS (including rollback segments)

Data Consistency and TSPITR

TSPITR provides views that can detect any data relationships between objects that are in the tablespaces being recovered and objects in the rest of the database. TSPITR will not successfully complete unless these relationships are managed, either by removing or suspending the relationship, or by including the related object within the recovery set.

See Also: For more information see "Step 2: Research and Resolve Dependencies on the Primary Database" on page 13-8, and "TS_PITR_CHECK Does Not Check for Objects Owned by SYS" on page 13-4.

TSPITR Requirements

You must satisfy the following requirements before performing TSPITR.

- All files that comprise the recovery set tablespaces must be present in the recovery set, otherwise the export phase of TSPITR will fail (with the error message "1230: cannot make read only, file %s is offline").
- You must create the control file backup in the auxiliary set using the following SQL statement:

```
ALTER DATABASE BACKUP CONTROLFILE TO '<controlfile_name>'
```

This control file backup must be created at a later time than the backup that is being used. If it's not, then you may encounter an error message (ORA-01152, file 1 was not restored from a sufficiently old backup).

- You must have enough disk space to accommodate the clone database (for example, the auxiliary set and the recovery set).
- You must have enough real memory to start up the clone instance.

See Also: For more information, see "Step 4: Prepare the Parameter Files for the Clone Database" on page 13-12.

Performing Tablespace Point-In-Time Recovery

This section describes how to perform TSPITR, and includes the following steps:

- Step 1: Find Out if Objects Will be Lost when Performing TSPITR
- Step 2: Research and Resolve Dependencies on the Primary Database
- Step 3: Prepare the Primary Database for TSPITR
- Step 4: Prepare the Parameter Files for the Clone Database
- Step 5: Prepare Clone Database for TSPITR
- Step 6: Recover the Clone Database
- Step 7: Open the Clone Database
- Step 8: Prepare the Clone Database for Export
- Step 9: Export the Clone Database
- Step 10: Copy the Recovery Set Clone Files to the Primary Database
- Step 11: Import into the Primary Database
- Step 12: Prepare the Primary Database for Use
- Step 13: Back Up the Recovered Tablespaces in the Primary Database

Step 1: Find Out if Objects Will be Lost when Performing TSPITR

When TSPITR is performed on a tablespace, any objects created after the point to which TSPITR is being performed will be lost. To see which objects will be lost, query the TS_PITR_OBJECTS_TO_BE_DROPPED view on the primary database. The contents of the view are described in Table 13–1:

Table 13–1 TS_PITR_OBJECTS_TO_BE_DROPPED View

Column Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
NAME	NOT NULL	VARCHAR2(30)
CREATION_TIME	NOT NULL	DATE
TABLESPACE_NAME		VARCHAR2(30)

When querying TS_PITR_OBJECTS_TO_BE_DROPPED, you must supply all the elements of the date field, otherwise the default setting will be used. You should also use the to_char and to_date functions. For example, with a recovery set consisting of TS1 and TS2, and a recovery point in time of '1997-06-02:07:03:11', you should issue the following query:

```
SVRMGR1> select owner, name, tablespace_name,
      2> to_char(creation_time, 'YYYY-MM-DD:HH24:MI:SS'),
      3> from ts_pitr_objects_to_be_dropped
      4> where tablespace_name in ('TS1','TS2')
      5> and
      6> creation_time > to_date('97-JUN-02:07:03:11','YY-MON-  DD:HH24:MI:SS')
      7> order by tablespace_name, creation_time
      8> / The information you find in TS_PITR_OBJECTS_TO_BE_DROPPED and TS_PITR_CHECK
           can help you decide whether or not to perform TSPITR.
```

Step 2: Research and Resolve Dependencies on the Primary Database

You can use the TS_PITR_CHECK view to identify relationships between objects that overlap the recovery set boundaries. If this view returns rows when queried, you must investigate and correct the problem. TSPITR can proceed *only* when TS_PITR_CHECK view returns no rows. You should record all actions performed during this step so that you can retrace these relationships after completing TSPITR.

The TS_PITR_CHECK view will return rows unless you meet the following requirements:

- All partitions of a partitioned table should be fully contained in the recovery set. If you need to recover only 1 or more of the partitions of a partitioned table, convert them to stand-alone tables (see "Performing Partial TSPITR of Partitioned Tables" on page 13-16).
- The following must be fully contained in the recovery set:
  - Tables (and their indexes, partitioned or non-partitioned)
  - Clusters (and their indexes, partitioned or non-partitioned)
  - Primary key and foreign key relationships
  - All elements of a LOB (lob segment, lob index and lob locator)
- The following object types must not exist in the recovery set:
  - replicated master tables
  - tables with varray columns
  - tables with nested tables
  - tables with external Bfiles
  - snapshot logs
  - snapshot tables

Querying the TS_PITR_CHECK View

Table 13–2 describes the contents of the TS_PITR_CHECK view.

Table 13–2 TS_PITR_CHECK View

Column	Datatype	NULL	Description
OBJ1_OWNER	VARCHAR2(30)	NOT NULL	The owner of the object preventing tablespace point-in-time recovery (see the REASON column for details)
OBJ1_NAME	VARCHAR2(30)	NOT NULL	The name of the object preventing tablespace point-in-time recovery
OBJ1_TYPE	VARCHAR2(15)		The object type for the object preventing tablespace point-in-time recovery
OBJ1_SUBNAME	VARCHAR2(30)		Subordinate to OBJ1_NAME

Table 13–2 TS_PITR_CHECK View (Cont.)

Column	Datatype	NULL	Description
TS1_NAME	VARCHAR2(30)	NOT NULL	Name of the tablespace containing the object preventing tablespace point-in-time recovery
OBJ2_NAME	VARCHAR2(30)		The name of a second object which may be preventing tablespace point-in-time recovery (if NULL, object 1 is the only object preventing recovery)
OBJ2_TYPE	VARCHAR2(15)		The object type for the second object (will be NULL if OBJ2_NAME is NULL)
OBJ2_OWNER	VARCHAR2(30)		The owner of the second object (will be NULL if OBJ2_NAME is NULL)
OBJ2_SUBNAME	VARCHAR2(30)		Subordinate to OBJ2_NAME
TS2_NAME	VARCHAR2(30)		Name of the tablespace containing second object that may be preventing tablespace point-in-time recovery (-1 indicates not applicable)
CONSTRAINT_NAME	VARCHAR2(30)		Name of the constraint
REASON	VARCHAR2(78)		Reason why tablespace point-in-time recovery cannot proceed

You must supply a four-line predicate detailing the recovery set tablespace to query the TS_PITR_CHECK view. For example, with a recovery set consisting of TS1 and TS2, the SELECT statement against TS_PITR_CHECK would be as follows:

```
SVRMGR 1> SELECT *
       2> FROM sys.ts_pitr_check
       3>   WHERE
       4>    (ts1_name in ('TS1','TS2')
       5>   AND ts2_name not in ('TS1','TS2'))
       6>   OR (ts1_name not in ('TS1','TS2')
       7>   AND ts2_name in ('TS1','TS2'))
       8> /
```

Due to the number and width of the columns in the TS_PITR_CHECK view, you may want to format the columns as follows:

```
column OBJ1_OWNER heading "owner1"
column OBJ1_OWNER format a6
column OBJ1_NAME heading "name1"
column OBJ1_NAME format a5
```

```
column OBJ1_SUBNAME heading "subname1"
column OBJ1_SUBNAME format a8
column OBJ1_TYPE heading "obj1type"
column OBJ1_TYPE format a8 word_wrapped
column TS1_NAME heading "ts1_name"
column TS1_NAME format a8
column OBJ2_NAME heading "name2"
column OBJ2_NAME format a5
column OBJ2_SUBNAME heading "subname2"
column OBJ2_SUBNAME format a8
column OBJ2_TYPE heading "obj2type"
column OBJ2_TYPE format a8 word_wrapped
column OBJ2_OWNER heading "owner2"
column OBJ2_OWNER format a6
column TS2_NAME heading "ts2_name"
column TS2_NAME format a8
column CONSTRAINT_NAME heading "cname"
column CONSTRAINT_NAME format a5
column REASON heading "reason"
column REASON format a57 word_wrapped
```

Sample Output If the partitioned table TP has two partitions, P1 and P2, which exist in tablespaces TS1 and TS2 respectively, and there is a partitioned index defined on TP called TPIND, which has two partitions ID1 and ID2 (that exist in tablespaces ID1 and ID2 respectively) you would get the following output when TS_PITR_CHECK is queried against tablespaces TS1 and TS2 (assuming appropriate formatting):

```
owner1  name1  subname1   obj1type   ts1_name   name2  subname2  obj2type  owner2   ts2_name  cname
reason
-----  ----   -----     ------   -------   ----   ------  --------  -------  ------  -----
SYSTEM  TP     P1        TABLE     TS1       TPIND   IP1     INDEX     PARTITION PARTITION  SYS ID1
Partitioned Objects not fully contained in the recovery set

SYSTEM   TP     P1       TABLE      TS1      TPIND IP2      INDEX    PARTITION PARTITION SYS ID2
Partitioned Objects not fully contained in the recovery set
```

You can see here that the table SYSTEM.TP has a partitioned index TPIND, which consists of two partitions, IP1 in tablespace ID1 and IP2 in tablespace ID2. Thus, you must decide to either drop TPIND or include ID1 and ID2 in the recovery set.

Step 3: Prepare the Primary Database for TSPITR

To prepare the primary database for TSPITR, perform the following tasks:

1. Issue the following statement on the primary database:

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

2. Take offline any rollback segments in the recovery set (you do not have to take auxiliary set rollback segments offline) using the following statement:

```
ALTER ROLLBACK SEGMENT <segment name> OFFLINE;
```

3. Take offline immediate the recovery set tablespaces on the primary database using the following statement:

```
ALTER TABLESPACE <tablespace name> OFFLINE IMMEDIATE;
```

   This prevents changes being made to the recovery set before TSPITR is complete.

> **Note:** If there is a subset of data (that is not physically or logically corrupt) you want to query within the recovery set tablespaces, you can alter the recovery set tablespaces on the primary database as READ ONLY for the duration of the recovery of the clone (this allows them to be queried but not altered). The recovery set tablespaces must be taken offline before integrating the clone files with the primary database (see "Step 10: Copy the Recovery Set Clone Files to the Primary Database" on page 13-15).

Step 4: Prepare the Parameter Files for the Clone Database

Create the parameter file from a new init.ora file (rather than using the file from the production instance); you can save memory by using "small" settings for parameters like DB_BLOCK_BUFFERS, SHARED_POOL_SIZE, or LARGE_POOL_SIZE. However, if the production parameter files are used for the clone database, it's possible that reducing these parameters would prevent the clone database from starting up when other parameters are set too high (such as the parameter ENQUEUE_RESOURCES, which allocates memory from within the shared pool). You must change the following parameters:

- CONTROL_FILES must point to the name and location of the clone control files

- LOCK_NAME_SPACE must be set to a unique value, for example, lock_name_space=CLONE

> **Note:** LOCK_NAME_SPACE allows the clone database to start up even though it has the same name as the primary database (otherwise it would fail because the mount lock is hashed from the database name). You should not change the DB_NAME parameter.

Change the following parameters if required:

- DB_FILE_NAME_CONVERT
- LOG_FILE_NAME_CONVERT

These parameters are used to update the clone database control file with the locations of the clone database files.

For example, if the datafiles of the primary database reside in the directory /ora/primary, and the clone will reside in the directory /ora/clone, then the value of DB_FILE_NAME_CONVERT should be set to “primary”,”clone”.

Step 5: Prepare Clone Database for TSPITR

Perform the following tasks to prepare the clone database for TSPITR:

1. Restore the auxiliary set and the recovery set to a location different from that of the primary database.

> **Note:** It is possible, although not recommended, to place the recovery set files over their corresponding files on the primary database. For more information see “Performing Partial TSPITR of Partitioned Tables” on page 13-6.

2. Set up your environment so that you can start up the clone database (for example, on UNIX, set ORACLE_SID to the name of the clone).
3. Startup nomount the clone database, specifying pfile if necessary (for example,’STARTUP NOMOUNT PFILE=/path/initCLONE.ora).
4. Mount the clone database using the following statement:

```
ALTER DATABASE MOUNT CLONE DATABASE;
```

At this point, the database is automatically taken out of archivelog mode because it is a clone. All files are offline at this point as well. Even if the file name conversion parameters DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT have been set, you cannot assume that all the files of the clone database will be in the locations specified by these parameters—there may be some clone database files that have been restored to different locations due to constraints of disk space. Additionally, the only files necessary to the clone database are in the recovery set and the auxiliary set; there may be many other database files that do not fall into these two sets that you can leave offline.

5. If DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT have not been set, you must use the following statement to rename the files to reflect their new locations:

```
ALTER DATABASE RENAME FILE '<name of file in primary location>'
  TO '<name of corresponding file in clone location>';
```

6. If DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT have been set, but there are files that have been restored to different locations, then they must be renamed at this point. Bring online all recovery set and auxiliary set files using the following SQL statement:

```
ALTER DATABASE DATAFILE '<datafile name>' ONLINE
;
```

Note: the export phase of TSPITR will not work if all the files of each recovery set tablespace are not online.

Step 6: Recover the Clone Database

Recover the clone database up to the desired point by specifying the USING BACKUP CONTROLFILE option. You can use any form of interrupted recovery, including time-based or cancel-based recovery, as follows:

```
RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL TIME 'YYYY-MM-DD:HH24:MI:SS';

RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL;
```

If the clone database files are not online you will get an error message ('ORA 264: no recovery required').

Step 7: Open the Clone Database

Alter the clone database open resetlogs using the following statement:

```
ALTER DATABASE OPEN RESETLOGS;
```

Because the database is a clone database, only the SYSTEM rollback segment is brought online at this point; this prevents you from executing DML statements against any user tablespace. Any attempt to bring a user rollback segment online will fail (message ORA 1698'a clone database may only have SYSTEM rollback segment online').

Step 8: Prepare the Clone Database for Export

Prepare the clone database for export using the TS_PITR_CHECK view and resolving the dependencies just as you did for the primary database (see "Step 2: Research and Resolve Dependencies on the Primary Database" on page 13-8). Only when TS_PITR_CHECK returns no rows will the export phase of TSPITR complete.

Step 9: Export the Clone Database

Export the meta-data for the recovery set tablespaces using the following statement:

```
exp sys/<password> point_in_time_recover=y
recovery_tablespaces=<tablespace1>,<tablespace2>,<tablespaceN>
```

If the export phase fails (with the error message "ORA 29308 view TS_PITR_CHECK failure"), re-query TS_PITR_CHECK, resolve the problem, and re-run the export. You need to perform the export phase of TSPITR as the user SYS, otherwise the export will fail (with the error message "ORA-29303: user does not login as SYS"). Shut down the clone database after a successful export.

Step 10: Copy the Recovery Set Clone Files to the Primary Database

If any recovery set tablespaces are READ ONLY on the primary database, you should change them to OFFLINE. Copy the recovery set files from the clone database to the primary database, taking care not to overwrite any auxiliary set files on the primary database.

Step 11: Import into the Primary Database

Import the recovery set meta-data into the primary database using the following command:

```
imp sys/<password> point_in_time_recover=true
```

This import also updates the copied file's file headers and integrates them with the primary database.

Step 12: Prepare the Primary Database for Use

First bring the recovery set tablespaces online in the primary database. Then change the recovery set tablespaces to READ WRITE (if they had been altered to READ ONLY, see "Step 3: Prepare the Primary Database for TSPITR" on page 13-12).

To prepare the primary database for use, undo all the steps taken to resolve dependencies; for example, rebuild indexes or re-enable constraints (see "Step 2: Research and Resolve Dependencies on the Primary Database" on page 13-8). If statistics existed on the recovery set objects before TSPITR was performed, you will need to recalculate them. For partitioned tables, you have to exchange the stand-alone tables into the partitions of their partitioned tables (for more information, see "Performing Partial TSPITR of Partitioned Tables" on page 13-16).

Step 13: Back Up the Recovered Tablespaces in the Primary Database

After TSPITR on a tablespace is complete, back up the tablespace.

> **WARNING: it is critical that you back up the tablespace. Failure to do so could result in the loss of that tablespace. For example, you could lose the tablespace in the event of media failure because the archived logs from the last backup of that database do not logically link to the recovered tablespaces. If you attempt to recover any recovery set tablespaces from a backup taken before TSPITR, you will fail (and receive error message "ORA 1246, recovering files through TSPITR of tablespace %s").**

Performing Partial TSPITR of Partitioned Tables

This section describes how to perform partial TSPITR of partitioned tables that have a range that has not changed or expanded, and includes the following steps:

- Step 1: Create a Table on the Primary Database for Each Partition Being Recovered
- Step 2: Drop the Indexes on the Partition Being Recovered
- Step 3: Exchange Partitions with Stand-Alone Tables
- Step 4:Take the Recovery Set Tablespace Offline

- Step 5: Create Tables at Clone Database
- Step 6: Drop Indexes on Partitions Being Recovered
- Step 7: Exchange Partitions with Stand-Alone Tables
- Step 8: Export the Clone Database
- Step 9: Copy the Recovery Set Datafiles to the Primary Database
- Step 10: Import into the Primary Database
- Step 11: Bring Recovery Set Tablespace Online
- Step 12: Exchange Partitions with Stand-Alone Tables
- Step 13: Back Up the Recovered Tablespaces in the Primary Database

> **Note:** Often, along with recovering a partition whose range has expanded, you will also have to recover the dropped partition. If this is the case, see "Performing TSPITR of Partitioned Tables When a Partition Has Been Dropped" on page 13-21.

Step 1: Create a Table on the Primary Database for Each Partition Being Recovered

Create a table on the primary database for each partition you wish to recover. This table should have the exact same column names and column datatypes as the partitioned table you are recovering. You can create the table using the following statement:

```
CREATE TABLE <new table> AS SELECT * FROM <partitioned table>  where 1=2;
```

These tables will be used to swap each recovery set partition (see Step 3).

Step 2: Drop the Indexes on the Partition Being Recovered

Drop the indexes on the partition you wish to recover, or create identical, non-partitioned indexes that exist on the partition you wish to recover (on the table created in Step 1). If you drop the indexes on the partition being recovered you will also need to drop them on the clone database (see Step 6). You will also have to rebuild the indexes after TSPITR is complete.

Step 3: Exchange Partitions with Stand-Alone Tables

Exchange each partition in the recovery set with its associated stand-alone table (created in Step 1) by issuing the following command:

```
ALTER TABLE <table name> EXCHANGE PARTITION <partition name> WITH TABLE <table name>;
```

Step 4:Take the Recovery Set Tablespace Offline

On the primary database, take each recovery set tablespace offline by issuing the following statement:

```
ALTER TABLESPACE <tablespace name> OFFLINE IMMEDIATE;
```

This prevents any further changes to the recovery set tablespaces on the primary database.

Step 5: Create Tables at Clone Database

After recovering the clone and opening resetlogs, create a table that has the exact same column names and column datatypes as the partitioned table you are recovering—do this for each partition you wish to recover. These tables will be used later to swap each recovery set partition.

Step 6: Drop Indexes on Partitions Being Recovered

Drop the indexes on the partition you wish to recover, or create identical, non-partitioned indexes that exist on the partition you wish to recover (on the table created in Step 1).

Step 7: Exchange Partitions with Stand-Alone Tables

For each partition in the clone database recovery set, exchange the partitions with the stand-alone tables (created in Step 5) by issuing the following statement:

```
ALTER TABLE <partitioned_table_name> EXCHANGE PARTITION <partition_name> WITH TABLE
<table_name>;
```

Step 8: Export the Clone Database

Execute export against the clone database for the recovery set tablespaces using the following statement:

```
exp sys/<password> point_in_time_recover=y
recovery_tablespaces=<tablespace1>,<tablespace2>,<tablespaceN>
```

If the export phase fails (with the error message "ORA 29308 view TS_PITR_CHECK failure"), re-query TS_PITR_CHECK, resolve the problem, and re-run the export. You need to perform the export phase of TSPITR as the user SYS, otherwise the export will fail (with the error message "ORA-29303: user does not login as SYS"). Shut down the clone database after a successful export.

Step 9: Copy the Recovery Set Datafiles to the Primary Database

If any recovery set tablespaces are READ ONLY on the primary database, you should change them to OFFLINE. Copy the recovery set datafiles from the clone database to the primary database, taking care not to overwrite any auxiliary set files on the primary database.

Step 10: Import into the Primary Database

Import the recovery set meta-data into the primary database using the following command:

```
imp sys/<password> point_in_time_recover=true
```

This import also updates the copied file's file headers and integrates them with the primary database.

Step 11: Bring Recovery Set Tablespace Online

At the primary database, bring each recovery set tablespace online by issuing the following statement:

```
ALTER TABLESPACE <tablespace name> ONLINE;
```

Step 12: Exchange Partitions with Stand-Alone Tables

For each recovered partition on the primary database, swap its associated stand-alone table back in using the following statement:

```
ALTER TABLE <table name> EXCHANGE PARTITION <partition name> WITH TABLE <table name>;
```

If the associated indexes have been dropped, you must re-create them.

Step 13: Back Up the Recovered Tablespaces in the Primary Database

Back up the recovered tablespaces on the primary database. Failure to do so will result in loss of data in the event of media failure.

WARNING: It is critical that you back up the tablespace. Failure to do so could result in the loss of that tablespace. For example, you could lose the tablespace in the event of media failure because the archived logs from the last backup of that database do not logically link to the recovered tablespaces. If you attempt to recover any recovery set tablespaces from a backup taken before TSPITR, you will fail (and receive error message "ORA 1246, recovering files through TSPITR of tablespace %s").

Performing TSPITR of Partitioned Tables When a Partition Has Been Dropped

This section describes how to perform TSPITR on partitioned tables when a partition has been dropped, and includes the following steps:

- Step 1: Find the Low and High Range of the Partition that Was Dropped
- Step 2: Create a Temporary Table
- Step 3: Delete Records From Partitioned Table
- Step 4: Take Recovery Set Tablespaces Offline
- Step 5: Create Tables at Clone Database
- Step 6: Drop Indexes on Partitions Being Recovered
- Step 7: Exchange Partitions with Stand-Alone Tables
- Step 8: Export the Clone Database
- Step 9: Copy the Recovery Set Datafiles to the Primary Database
- Step 10: Import into the Primary Database
- Step 11: Bring Recovery Set Tablespace Online
- Step 12: Insert Stand-Alone Tables into Partitioned Tables
- Step 13: Back Up the Recovered Tablespaces in the Primary Database

Step 1: Find the Low and High Range of the Partition that Was Dropped

When a partition is dropped, the range of the partition above it expands downwards. Therefore, there may be records in the partition above that should actually be in the dropped partition after it has been recovered. To ascertain this, issue the following command at the primary database:

```
SELECT * FROM <partitioned table> WHERE <relevant key> BETWEEN <low range of partition
that was dropped> and <high range of partition that was dropped>;
```

Step 2: Create a Temporary Table

If any records are returned, create a temporary table in which to store these records so that they can be inserted into the recovered partition later (if required).

Step 3: Delete Records From Partitioned Table

Delete all the records stored in the temporary table from the partitioned table.

Step 4: Take Recovery Set Tablespaces Offline

At the primary database, take each recovery set tablespace offline by issuing the following statement:

```
ALTER TABLESPACE <tablespace name> OFFLINE IMMEDIATE;
```

Step 5: Create Tables at Clone Database

After recovering the clone and opening resetlogs, create a table that has the exact same column names and column datatypes as the partitioned table you are recovering—do this for each partition you wish to recover. These tables will be used later to swap each recovery set partition.

Step 6: Drop Indexes on Partitions Being Recovered

Drop the indexes on the partition you wish to recover, or create identical, non-partitioned indexes that exist on the partition you wish to recover.

Step 7: Exchange Partitions with Stand-Alone Tables

For each partition in the clone recovery set, exchange the partitions into the stand-alone tables created in Step 5 by issuing the following statement:

```
ALTER TABLE <partitioned_table_name> EXCHANGE PARTITION <partition_name> WITH TABLE
<table_name>;
```

Step 8: Export the Clone Database

Execute export against the clone database for the recovery set tablespaces using the following statement:

```
exp sys/<password> point_in_time_recover=y
recovery_tablespaces=<tablespace1>,<tablespace2>,<tablespaceN>
```

If the export phase fails (with the error message “ORA 29308 view TS_PITR_CHECK failure”), re-query TS_PITR_CHECK, resolve the problem, and

re-run the export. You need to perform the export phase of TSPITR as the user SYS, otherwise the export will fail (with the error message "ORA-29303: user does not login as SYS"). Shut down the clone database after a successful export.

Step 9: Copy the Recovery Set Datafiles to the Primary Database

If any recovery set tablespaces are READ ONLY on the primary database, you should change them to OFFLINE. Copy the recovery set datafiles from the clone database to the primary database, taking care not to overwrite any auxiliary set files on the primary database.

Step 10: Import into the Primary Database

Import the recovery set meta-data into the primary database using the following command:

```
imp sys/<password> point_in_time_recover=true
```

This import also updates the copied file's file headers and integrates them with the primary database.

Step 11: Bring Recovery Set Tablespace Online

Online each recovery set tablespace at the primary database by issuing the following statement:

```
ALTER TABLESPACE <tablespace name> ONLINE;
```

Step 12: Insert Stand-Alone Tables into Partitioned Tables

At this point you must insert the stand-alone tables into the partitioned tables; you can do this by first issuing the following statement:

```
ALTER TABLE <table name> SPLIT PARTITION <partition name> AT (<key value>) INTO
      (PARTITION <partition 1 name> TABLESPACE <tablespace name>, PARTITION <partition 2
       name> TABLESPACE <tablespace name>);
```

Note that at this point, partition 2 is empty because keys in that range have already been deleted from the table.

Issue the following statement to swap the stand-alone table into the partition:

```
 ALTER TABLE EXCHANGE PARTITION <partition name> WITH TABLE <table name>;
```

Now insert the records saved in Step 2 into the recovered partition (if desired).

Note: If the partition that has been dropped is the last partition in the table, it can be added using the following statement:

```
ALTER TABLE ADD PARTITION;
```

Step 13: Back Up the Recovered Tablespaces in the Primary Database

Back up the recovered tablespaces in the primary database. Failure to do so will result in loss of data in the event of media failure.

WARNING: It is critical that you back up the tablespace. Failure to do so could result in the loss of that tablespace. For example, you could lose the tablespace in the event of media failure because the archived logs from the last backup of that database do not logically link to the recovered tablespaces. If you attempt to recover any recovery set tablespaces from a backup taken before TSPITR, you will fail (and receive error message "ORA 1246, recovering files through TSPITR of tablespace %s").

Note: As described in "Limitations Advisory" on page 13-4, TSPITR cannot be used to recover a tablespace that has been dropped. Therefore, if the associated tablespace of the partition has been dropped as well as the partition, you cannot recover that partition using TSPITR. You will have to perform ordinary export/import recovery. Specifically, you will have to:

- Make a copy of the database
- Roll it forward
- Open the database
- Exchange the partition for a stand-alone table
- Make a table-level export of the stand-alone table
- Import the table into the primary database and insert it into the partitioned table using the following statement:

```
ALTER TABLE SPLIT PARTITION or ALTER TABLE ADD
PARTITION;
```

Performing TSPITR of Partitioned Tables When a Partition Has Been Split

This section describes how to recover partitioned tables when a partition has been split, and includes the following sections:

- Step 1: Drop the Lower of the Two Partitions at the Primary Database
- Step 2: Drop Indexes of Partitions Being Recovered
- Step 3: Exchange Partitions with Stand-Alone Tables
- Step 4: Take Recovery Set Tablespaces Offline
- Step 5: Create Tables at Clone Database
- Step 6: Drop Indexes in Partitions Being Recovered
- Step 7: Exchange Partitions with Stand-Alone Tables
- Step 8: Export the Clone Database
- Step 9: Copy the Recovery Set Datafiles to the Primary Database
- Step 10: Import into the Primary Database
- Step 11: Bring Recovery Set Tablespace Online
- Step 12: Exchange Partitions with Stand-Alone Tables
- Step 13: Back Up the Recovered Tablespaces in the Primary Database

Step 1: Drop the Lower of the Two Partitions at the Primary Database

At the primary database, for each partition you wish to recover whose range has been split, drop the lower of the two partitions so that the higher of the two partitions expands downwards (in other words, has the same range as before the split). For example, if P1 was split into two partitions P1A and P1B, then P1B must be dropped, meaning that partition P1A now has the same range as P1.

For each partition that you wish to recover whose range has split, create a table that has exactly the same column names and column datatypes as the partitioned table you are recovering by issuing the following statement:

```
CREATE TABLE <new table> AS SELECT * FROM <partitioned table> where 1=2;
```

These tables will be used to exchange each recovery set partition Step 3.

Step 2: Drop Indexes of Partitions Being Recovered

Either drop the indexes of the partition you wish to recover or create identical, non-partitioned indexes on the table created in Step 1 as the indexes that exist on the partition you wish to recover. Dropping the indexes on the partition you wish to recover means that you will need also to drop them an the clone database (see Step 6), and will of course mean that they need to be rebuilt once the recovery is complete.

Step 3: Exchange Partitions with Stand-Alone Tables

Exchange each partition in the recovery set with its associated stand-alone table (created in Step 1) by issuing the following command:

```
ALTER TABLE <table name> EXCHANGE PARTITION <partition name> WITH TABLE <table name>;
```

Step 4: Take Recovery Set Tablespaces Offline

At the primary database, take each recovery set tablespace offline by issuing the following statement:

```
ALTER TABLESPACE <tablespace name> OFFLINE IMMEDIATE;
```

This prevents any further changes to the recovery set tablespaces at the primary database.

Step 5: Create Tables at Clone Database

At the clone database, after recovering the clone and opening resetlogs: For each partition you wish to recover, create a table that has exactly the same column names and column datatypes as the partitioned table you are recovering. These tables will be used to swap each recovery set partition (see Step 7).

Step 6: Drop Indexes in Partitions Being Recovered

Either drop the indexes on the partition you wish to recover, or create identical, non-partitioned indexes on the table created in Step 1 as the indexes that exist on the partition you wish to recover.

Step 7: Exchange Partitions with Stand-Alone Tables

For each partition in the clone recovery set, exchange the partitions into the stand-alone tables created in Step 5 by issuing the following statement:

```
ALTER TABLE <partitioned_table_name> EXCHANGE PARTITION <partition_name> WITH TABLE
<table_name>;
```

Step 8: Export the Clone Database

Execute export against the clone database for the recovery set tablespaces using the following statement:

```
exp sys/<password> point_in_time_recover=y
recovery_tablespaces=<tablespace1>,<tablespace2>,<tablespaceN>
```

If the export phase fails (with the error message "ORA 29308 view TS_PITR_CHECK failure"), re-query TS_PITR_CHECK, resolve the problem, and re-run the export. You need to perform the export phase of TSPITR as the user SYS, otherwise the export will fail (with the error message "ORA-29303: user does not login as SYS"). Shut down the clone database after a successful export.

Step 9: Copy the Recovery Set Datafiles to the Primary Database

If any recovery set tablespaces are READ ONLY on the primary database, you should change them to OFFLINE. Copy the recovery set datafiles from the clone database to the primary database, taking care not to overwrite any auxiliary set files on the primary database.

Step 10: Import into the Primary Database

Import the recovery set meta-data into the primary database using the following command:

```
imp sys/<password> point_in_time_recover=true
```

This import also updates the copied file's file headers and integrates them with the primary database.

Step 11: Bring Recovery Set Tablespace Online

Bring each recovery set tablespace at the primary database online by issuing the following statement:

```
ALTER TABLESPACE <tablespace name> ONLINE;
```

Step 12: Exchange Partitions with Stand-Alone Tables

For each recovered partition at the primary database, exchange its associated stand-alone table using the following statement:

```
ALTER TABLE <table name> EXCHANGE PARTITION <partition name> WITH TABLE <table name>;
```

If the associated indexes have been dropped, you must re-create them.

Step 13: Back Up the Recovered Tablespaces in the Primary Database

Back up the recovered tablespaces in the primary database. Failure to do so will result in loss of data in the event of media failure.

See Also: For more information see "Performing TSPITR of Partitioned Tables When a Partition Has Been Dropped" on page 13-21.

WARNING: it is critical that you back up the tablespace. Failure to do so could result in the loss of that tablespace. For example, you could lose the tablespace in the event of media failure because the archived logs from the last backup of that database do not logically link to the recovered tablespaces. If you attempt to recover any recovery set tablespaces from a backup taken before TSPITR, you will fail (and receive error message "ORA 1246, recovering files through TSPITR of tablespace %s").

TSPITR Tuning Considerations

This section describes tuning issues relevant to TSPITR, and includes the following topics:

- Recovery Set Location Considerations
- Backup Control File Considerations

Recovery Set Location Considerations

If space is at a premium, it is possible to recover the recovery set files 'in place', in other words, over their corresponding files on the primary database. This is not the recommended best practice—the recommended best practice is that you restore the files to a separate location and then copy across before the import phase of TSPITR is complete (see "Step 11: Import into the Primary Database" on page 13-15).

Following are advantages and disadvantages of the two approaches.

Advantages and Disadvantages of Recovering to a Separate Location

An advantage of recovering to a separate location is basically greater availability and flexibility. If the recovery is abandoned at a point before integrating the recovery set with the primary database then there is no need to restore the recovery set files on the primary database and recover them using normal means. Also, the recovery set tablespaces can be accessible on the primary database while recovery occurs on the clone. For example, there may be a subset of undamaged data within

the recovery set tablespaces that you wish to access (see “Step 3: Prepare the Primary Database for TSPITR” on page 13-12). If this is the case, you can change the recovery set tablespaces to READ ONLY on the primary database so that you can query them while preventing any further changes to them. If the files are recovered in place this is not possible.

A disadvantage of recovering to s separate location is that more space is required for the clone database.

Advantages and Disadvantages of Recovering in Place

An advantage of recovering in place is that the amount of space taken up by the recovery set files is saved. After recovery of the clone is complete, there is no need to copy the recovery set files over to the primary database.

If the recovery is abandoned at a point before integrating the recovery set with the primary database (see “Step 11: Import into the Primary Database” on page 13-15) then the overwritten recovery set files of the primary database must be restored from a backup and recovered by normal means, prolonging data unavailability—this is a disadvantage. You cannot query any undamaged data within the recovery set tablespaces while recovery is going on.

Backup Control File Considerations

The error “ORA-01152 file 1 was not restored from a sufficiently old backup” will be encountered in the situation where no recovery is performed on the clone before grafting it to the primary. For example, if a backup is taken at time A, and a database at time B requires TSPITR to be done on a particular tablespace to take that tablespace to time A, what actually happens is that the clone database is opened resetlogs without any recovery having been done, i.e. when recovering the clone, the commands would be:

```
SVRMGRL> recover database using backup controlfile until cancel;
SVRMGRL> cancel;
SVRMGRL> open database resetlogs;
```

At this point no logs have been applied, but we wish to open the database. However, since we save checkpoints to the control file in Oracle 8, it is a requirement for clone and standby databases that the backup control files need to be taken at a point after the rest of the backup was taken. Unless this is the case, “ORA-01152 file 1 was not restored from a sufficiently old backup” will be

encountered on open, not because file 1 is too recent (because it is in sync with the rest of the database), but because it is more recent than the control file.

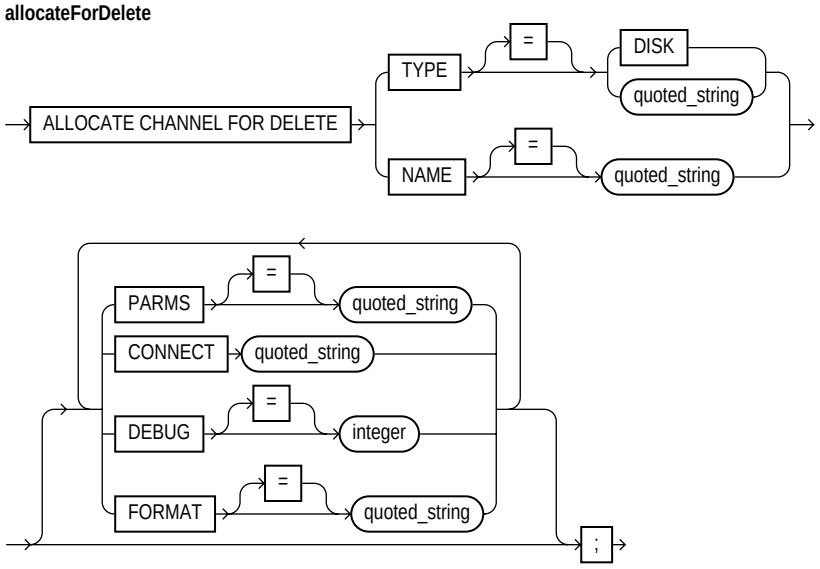
Note: A resetlogs would work with a regular database if a clean, consistent backup and an old backup control file is used, otherwise the behavior would not be compatible with existing backup scripts.

A

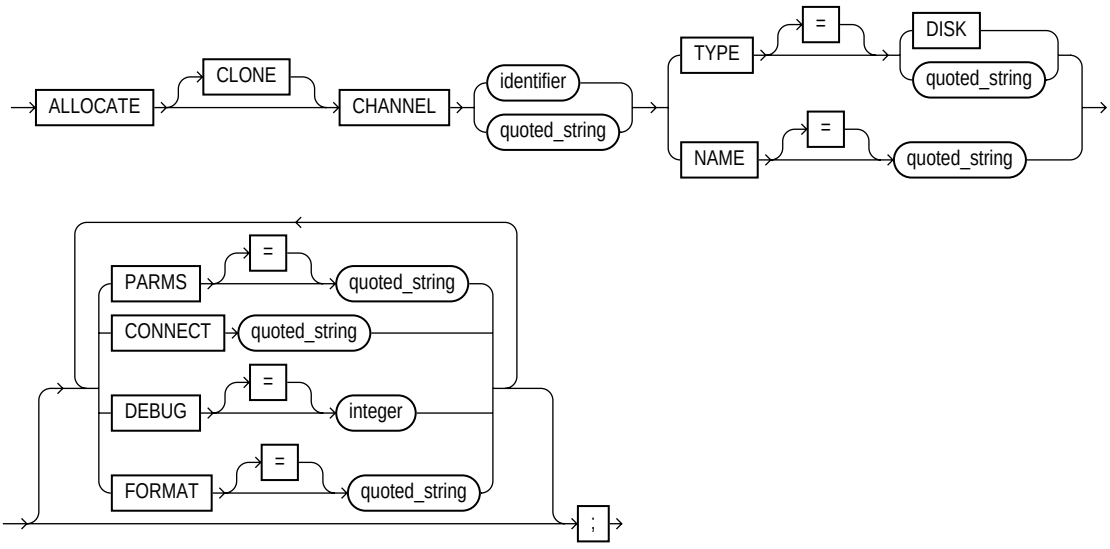
Recovery Manager Command Syntax

This appendix includes the Recovery Manager command syntax.

allocateForDelete

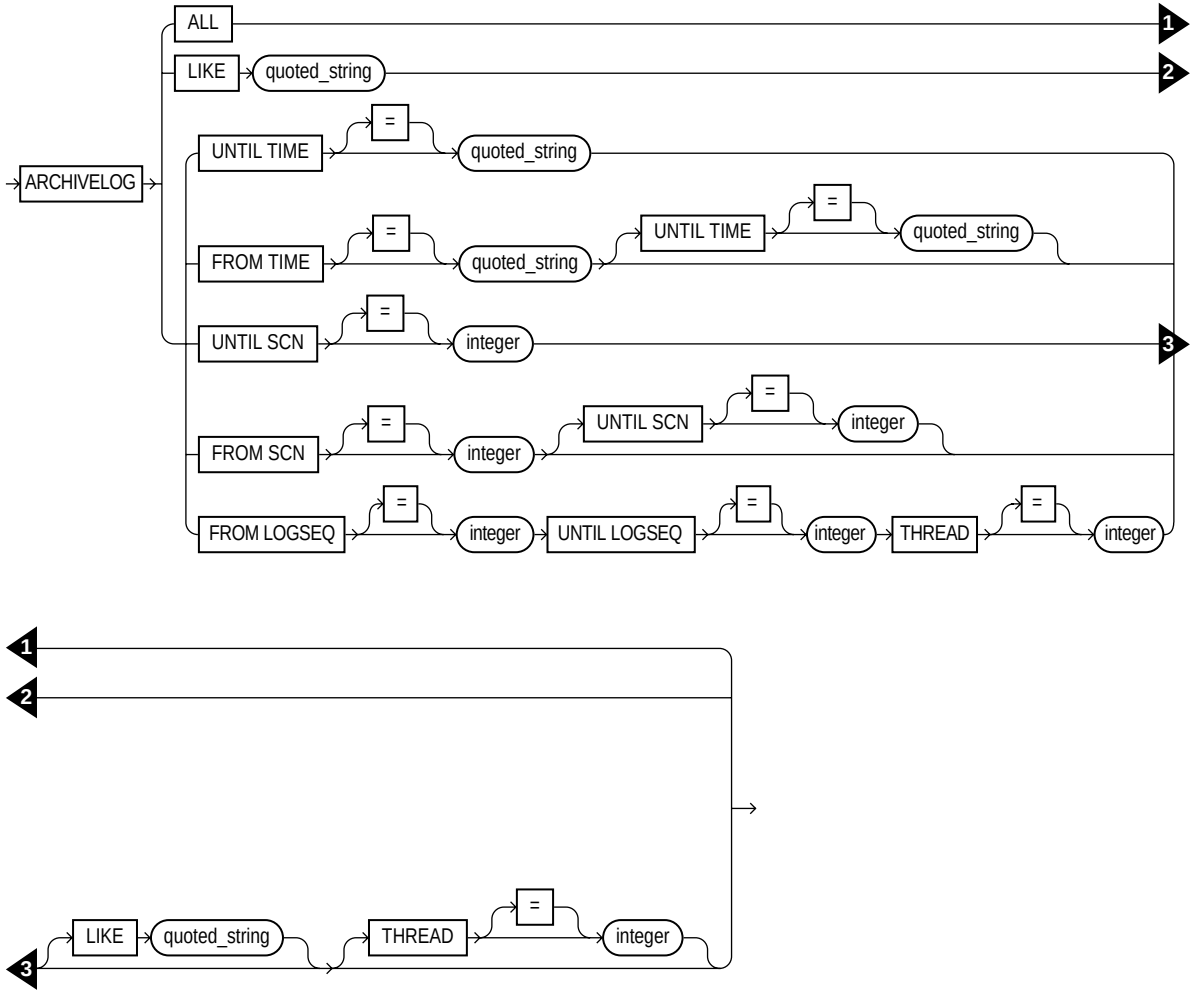


allocate


ALLOCATE
CLONE
CHANNEL
identifier
quoted_string
TYPE
=
DISK
quoted_string
NAME
=
quoted_string
PARMS
=
quoted_string
CONNECT
quoted_string
DEBUG
=
integer
FORMAT
=
quoted_string
;

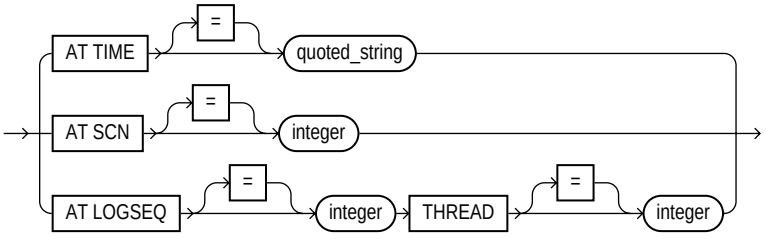
archivelogRecordSpecifier

archivelogRecordSpecifier

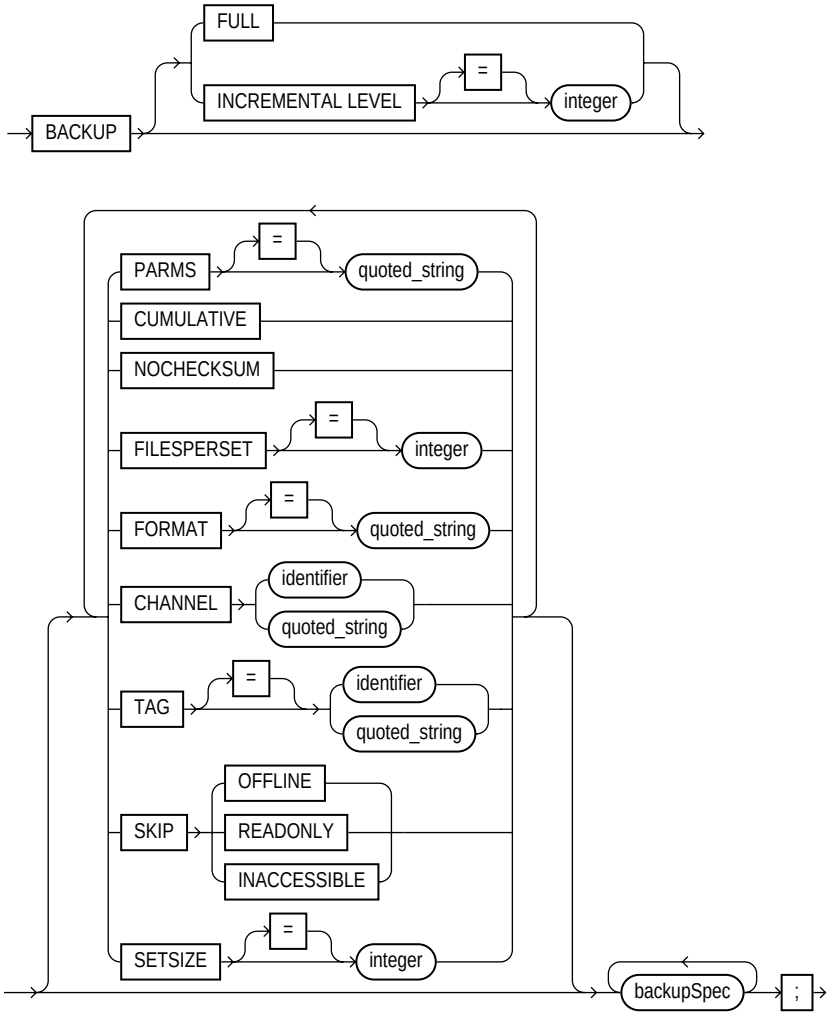


atClause

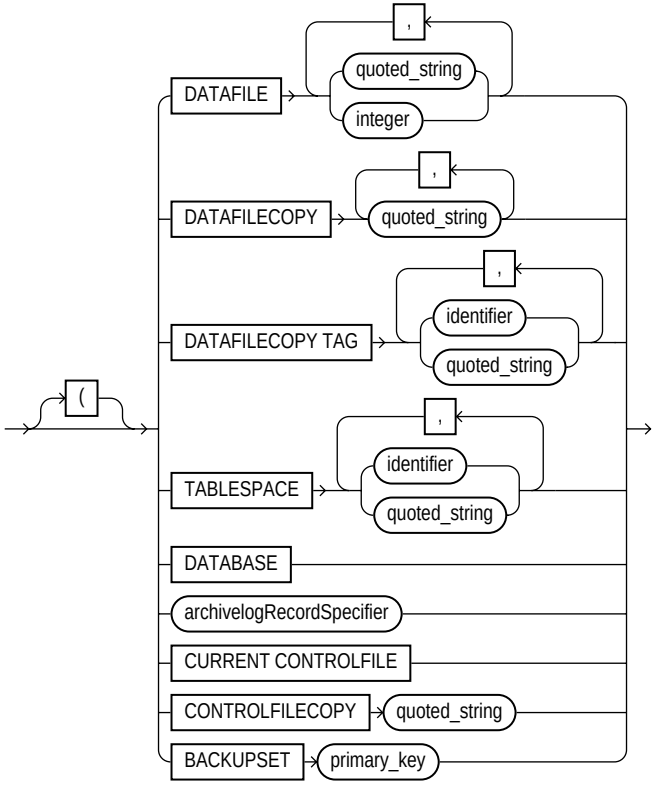
atClause

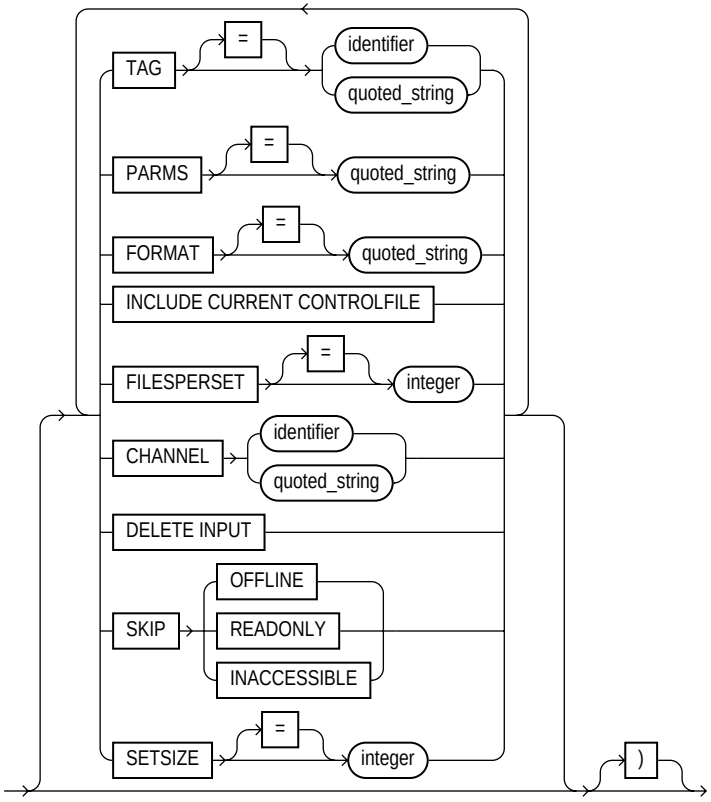


backup

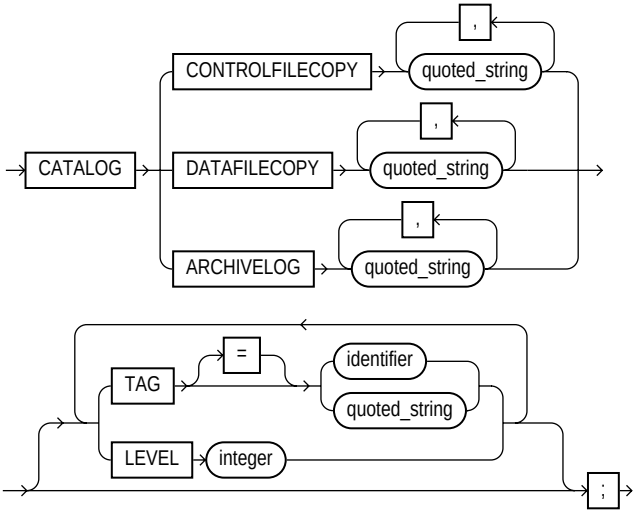

BACKUP
FULL
INCREMENTAL LEVEL
=
integer
PARMS
=
quoted_string
CUMULATIVE
NOCHECKSUM
FILESPERSET
=
integer
FORMAT
=
quoted_string
CHANNEL
identifier
quoted_string
TAG
=
identifier
quoted_string
SKIP
OFFLINE
READONLY
INACCESSIBLE
SETSIZE
=
integer
backupSpec
;

backupSpec

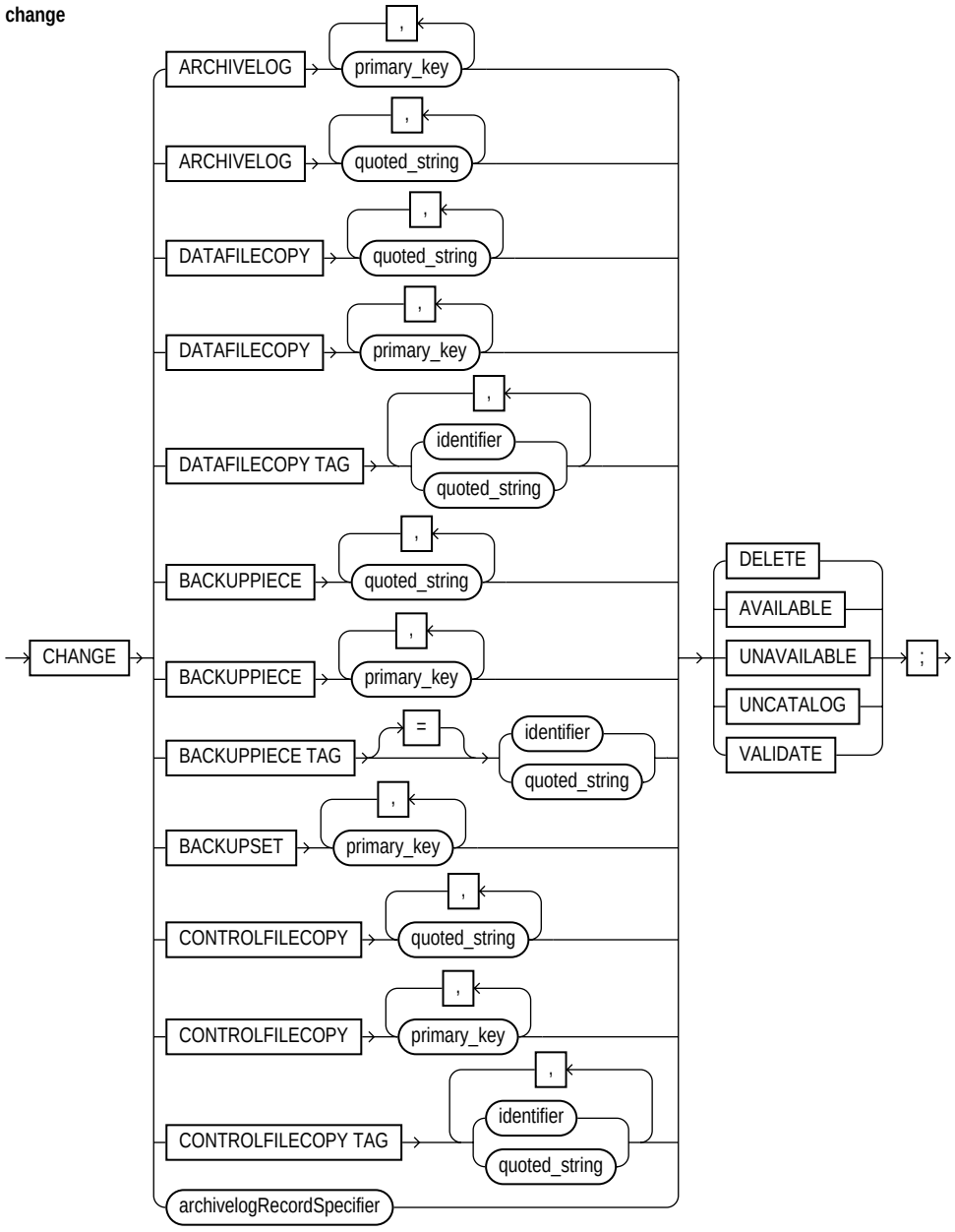

(
DATAFILE
quoted_string
integer
,
DATAFILECOPY
quoted_string
,
DATAFILECOPY TAG
identifier
quoted_string
,
TABLESPACE
identifier
quoted_string
,
DATABASE
archivelogRecordSpecifier
CURRENT CONTROLFILE
CONTROLFILECOPY
quoted_string
BACKUPSET
primary_key


TAG
=
identifier
quoted_string
PARMS
=
quoted_string
FORMAT
=
quoted_string
INCLUDE CURRENT CONTROLFILE
FILESPERSET
=
integer
CHANNEL
identifier
quoted_string
DELETE INPUT
SKIP
OFFLINE
READONLY
INACCESSIBLE
SETSIZE
=
integer
)

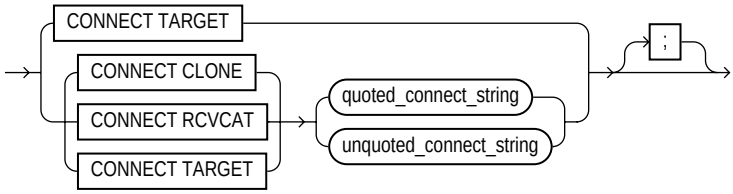
catalog


CATALOG
CONTROLFILECOPY
DATAFILECOPY
ARCHIVELOG
quoted_string
,
TAG
=
identifier
quoted_string
LEVEL
integer
;

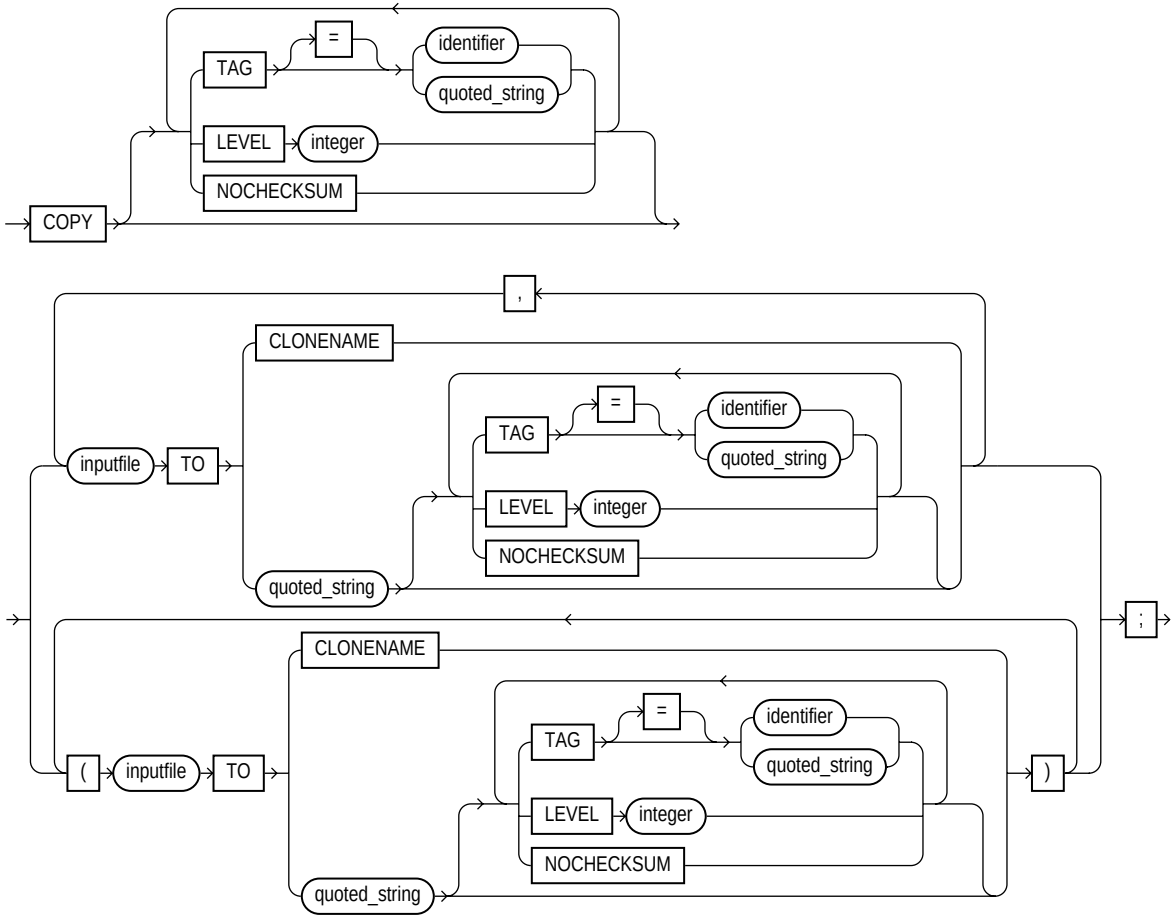
change


change
CHANGE
ARCHIVELOG
primary_key
ARCHIVELOG
quoted_string
DATAFILECOPY
quoted_string
DATAFILECOPY
primary_key
DATAFILECOPY TAG
identifier
quoted_string
BACKUPPIECE
quoted_string
BACKUPPIECE
primary_key
BACKUPPIECE TAG
=
identifier
quoted_string
BACKUPSET
primary_key
CONTROLFILECOPY
quoted_string
CONTROLFILECOPY
primary_key
CONTROLFILECOPY TAG
identifier
quoted_string
archivelogRecordSpecifier
DELETE
AVAILABLE
UNAVAILABLE
UNCATALOG
VALIDATE
;

connect

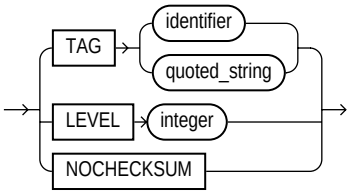

CONNECT TARGET
CONNECT CLONE
CONNECT RCVCAT
CONNECT TARGET
quoted_connect_string
unquoted_connect_string
;

copy

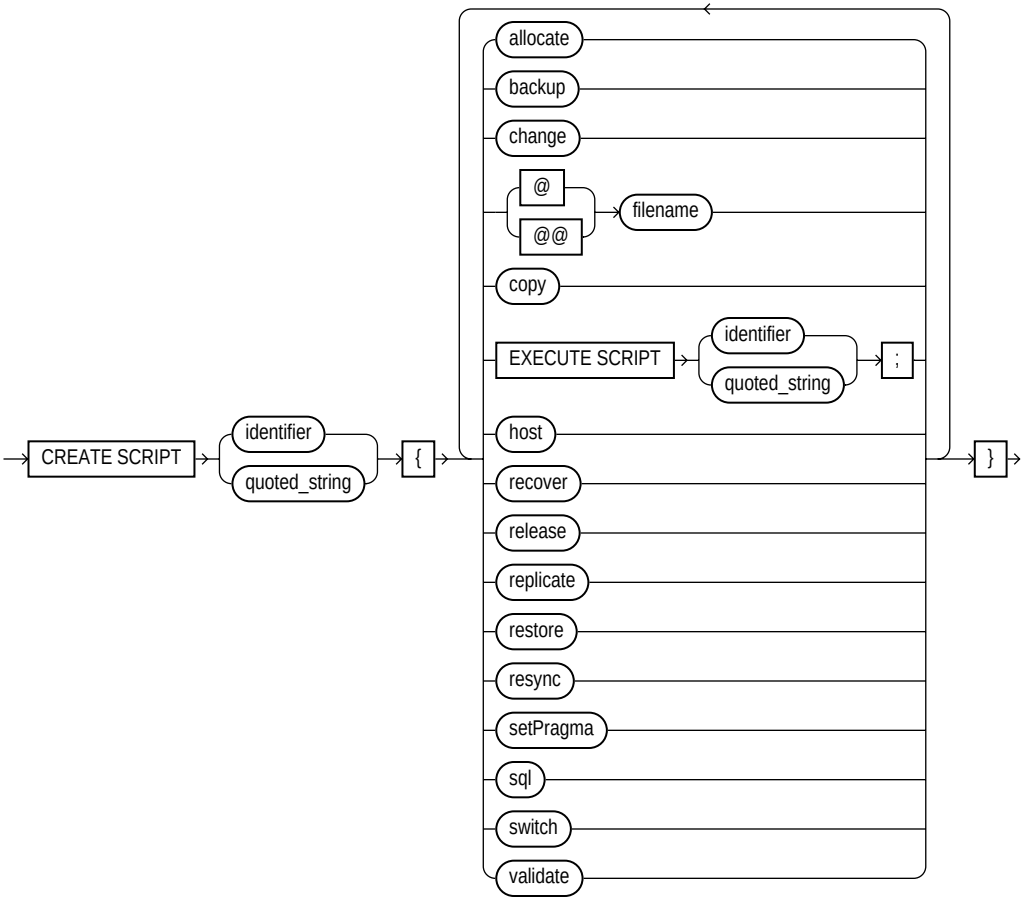

COPY
TAG
=
identifier
quoted_string
LEVEL
integer
NOCHECKSUM
inputfile
TO
CLONENAME
quoted_string
,
(
)
;

copyOption

copyOption

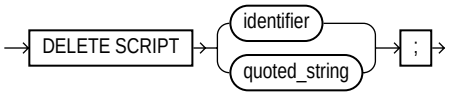


createScript



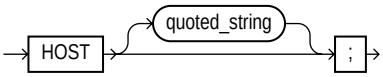
deleteScript

deleteScript



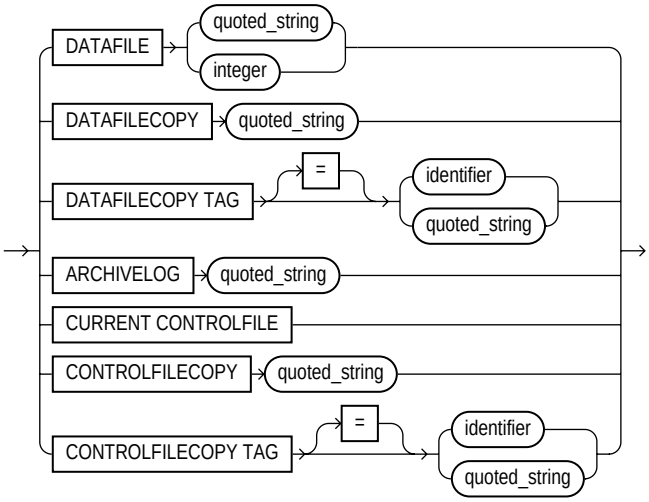
host

host

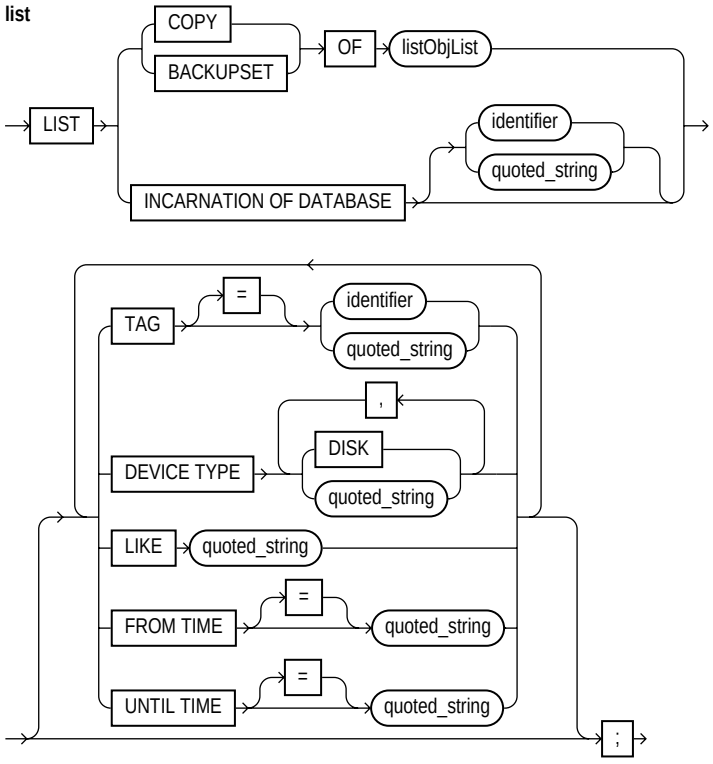


inputfile

inputfile

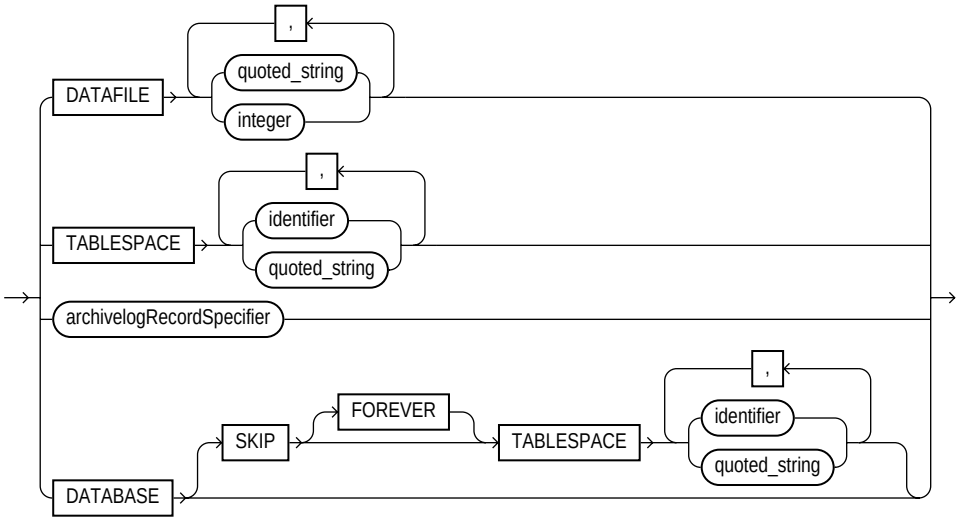


list


list
COPY
BACKUPSET
OF
listObjList
LIST
identifier
quoted_string
INCARNATION OF DATABASE
TAG
=
identifier
quoted_string
,
DISK
DEVICE TYPE
quoted_string
LIKE
quoted_string
FROM TIME
=
quoted_string
UNTIL TIME
=
quoted_string
;

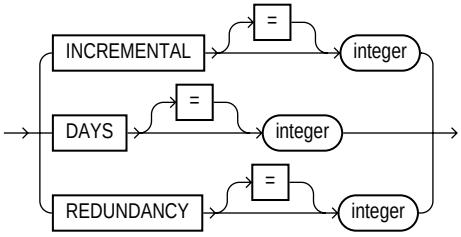
listObjList

listObjList



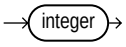
needBackupOperand

needBackupOperand



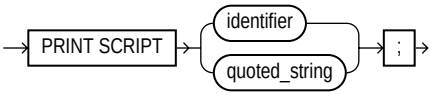
primary_key

primary_key

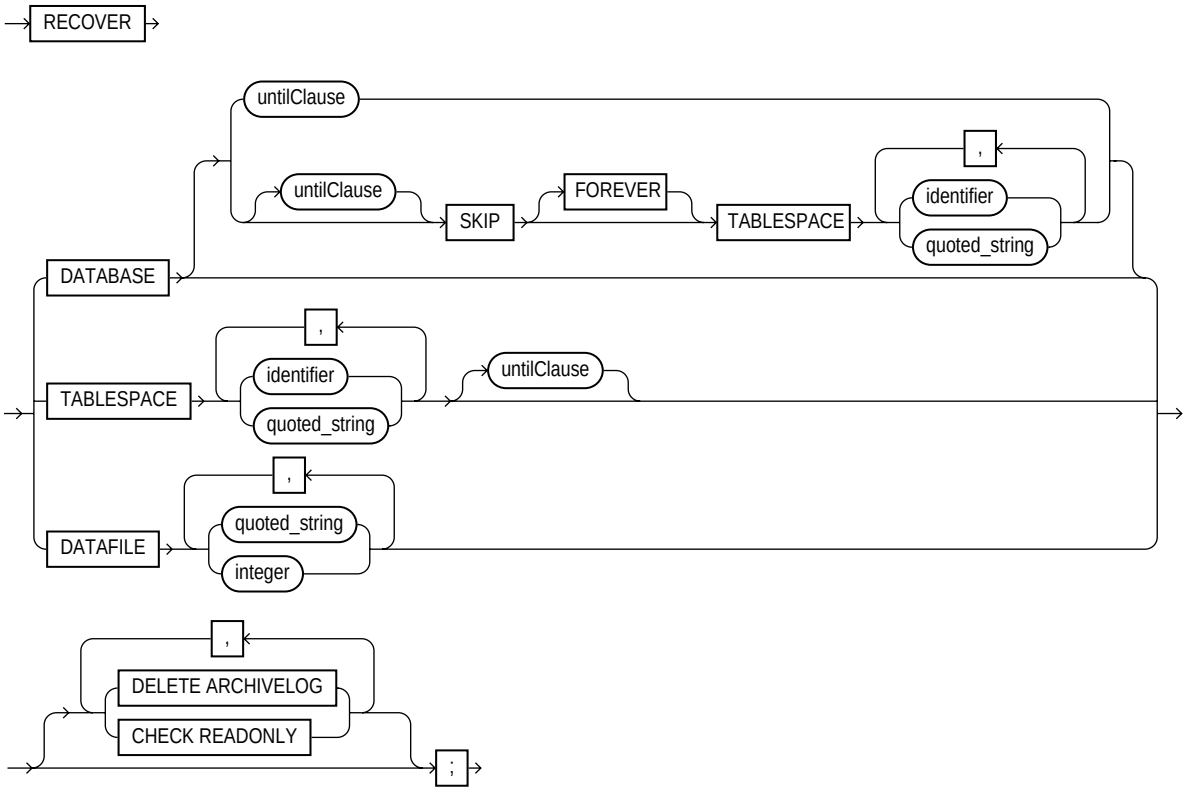


printScript

printScript



recover

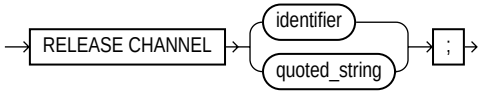

RECOVER
untilClause
untilClause
SKIP
FOREVER
TABLESPACE
,
identifier
quoted_string
DATABASE
TABLESPACE
,
identifier
quoted_string
untilClause
DATAFILE
,
quoted_string
integer
,
DELETE ARCHIVELOG
CHECK READONLY
;

register

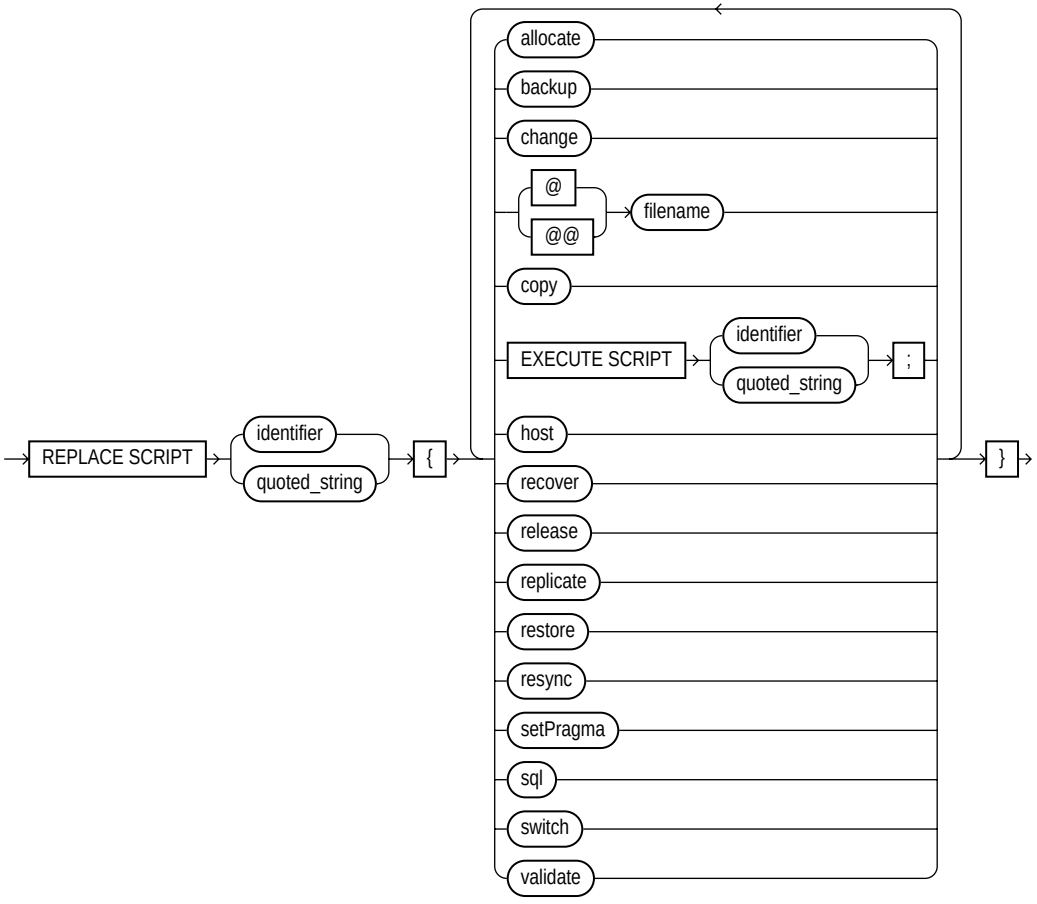
register

release

release



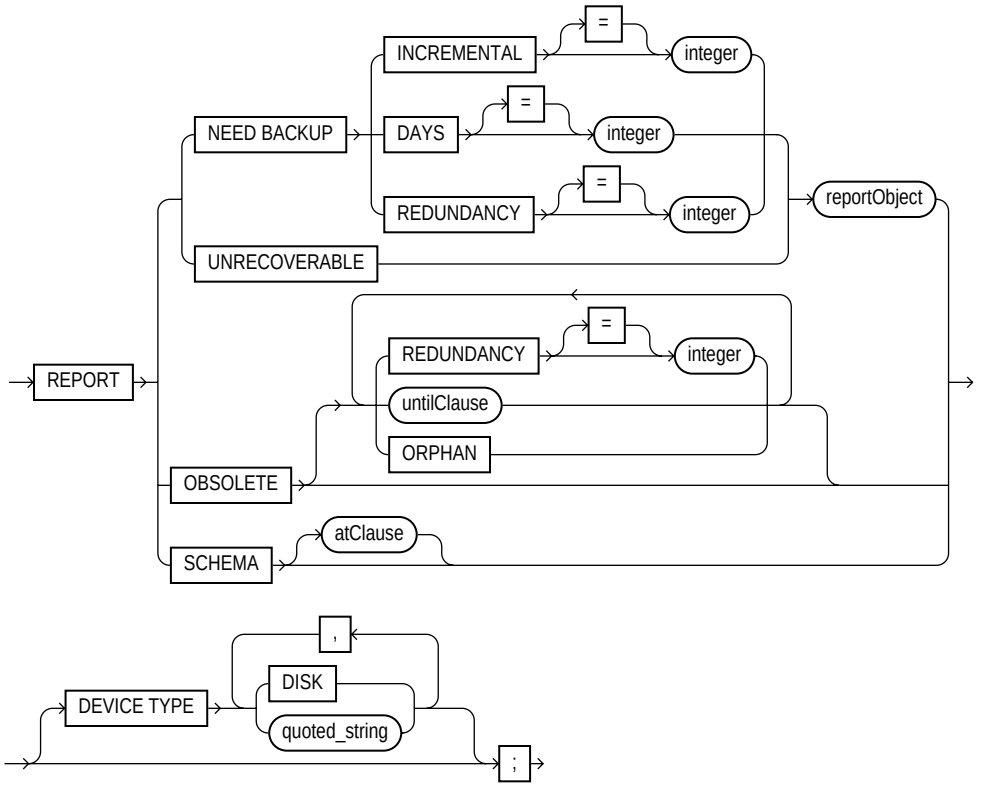
replaceScript



replicate

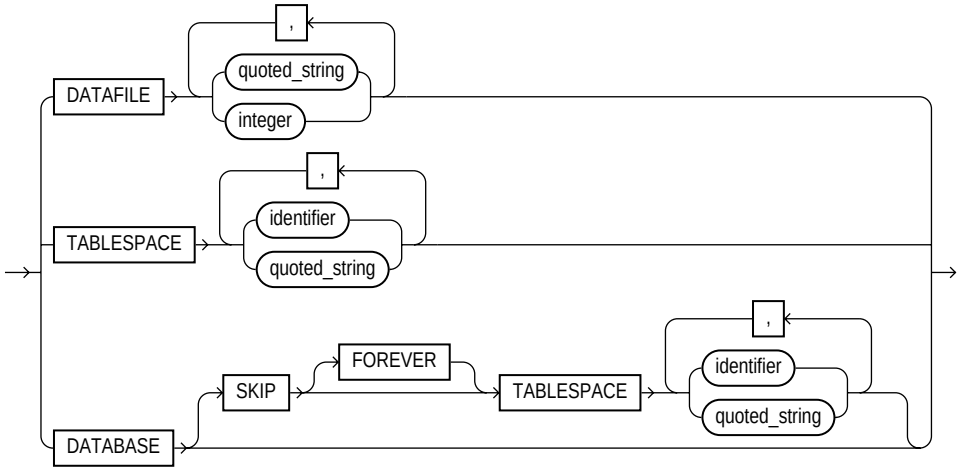
replicate

report


REPORT
NEED BACKUP
INCREMENTAL
=
integer
DAYS
=
integer
REDUNDANCY
=
integer
UNRECOVERABLE
reportObject
OBSOLETE
REDUNDANCY
=
integer
untilClause
ORPHAN
SCHEMA
atClause
DEVICE TYPE
,
DISK
quoted_string
;

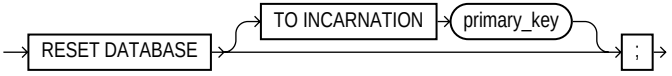
reportObject

reportObject



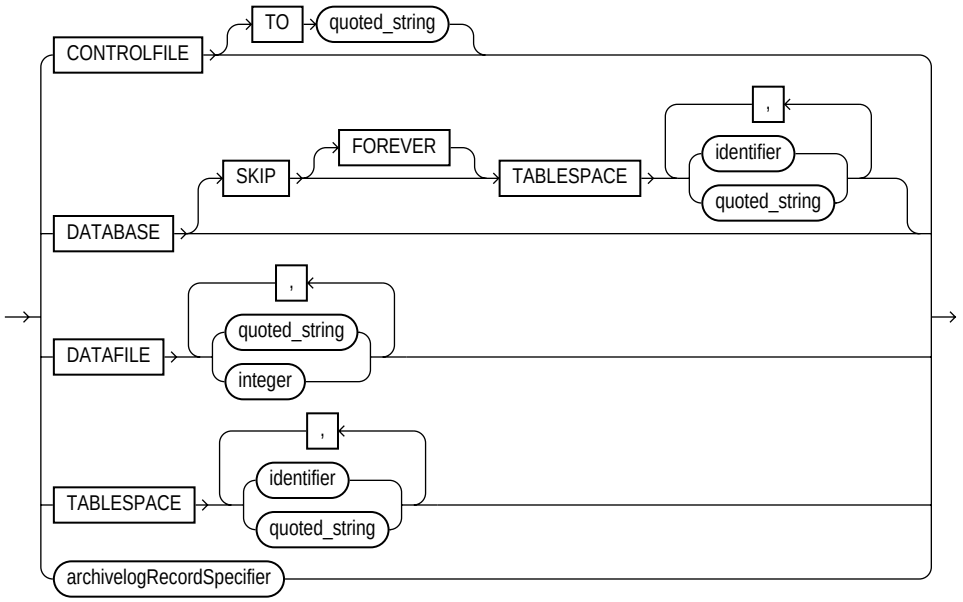
reset

reset

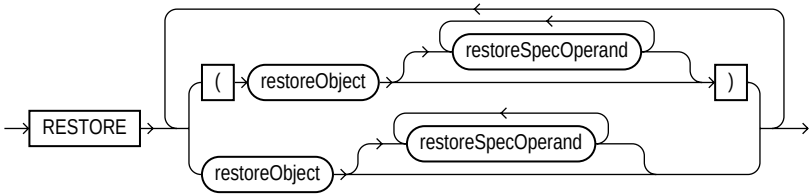


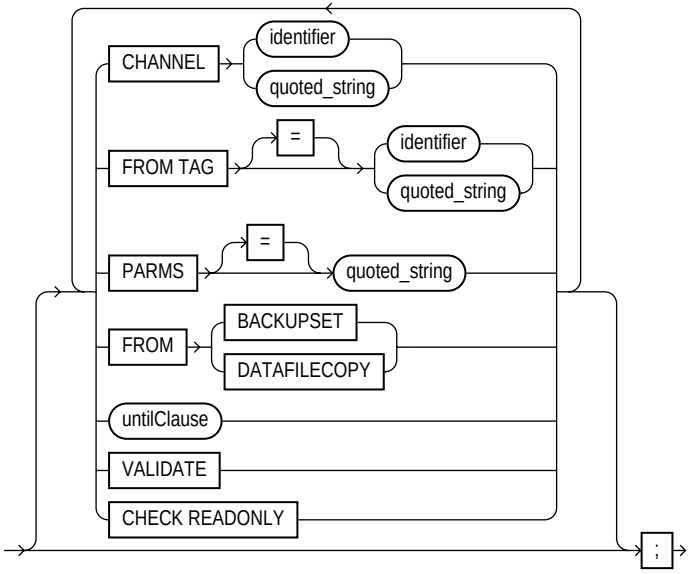
restoreObject

restoreObject



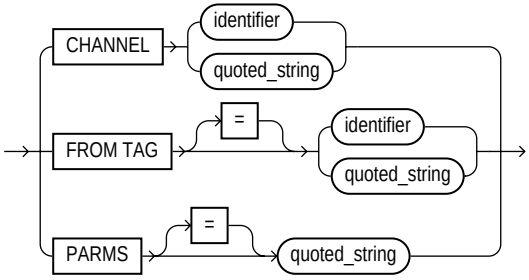
restore


RESTORE
(
restoreObject
restoreSpecOperand
)
restoreObject
restoreSpecOperand


CHANNEL
identifier
quoted_string
FROM TAG
=
identifier
quoted_string
PARMS
=
quoted_string
FROM
BACKUPSET
DATAFILECOPY
untilClause
VALIDATE
CHECK READONLY
;

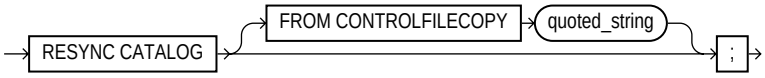
restoreSpecOperand

restoreSpecOperand



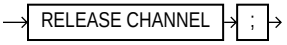
resync

resync

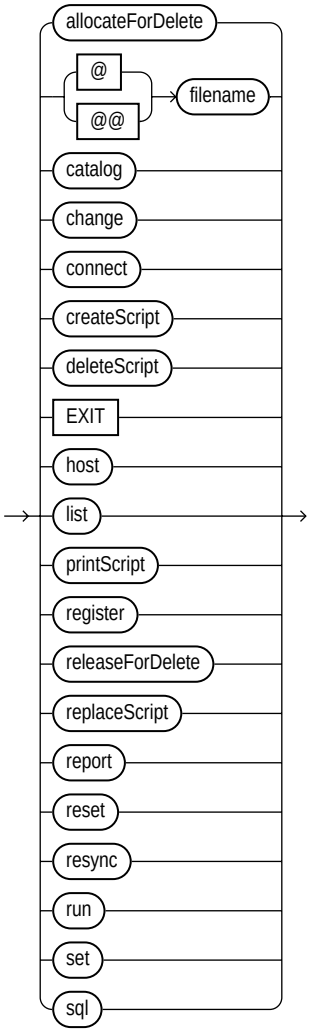


releaseForDelete

releaseForDelete

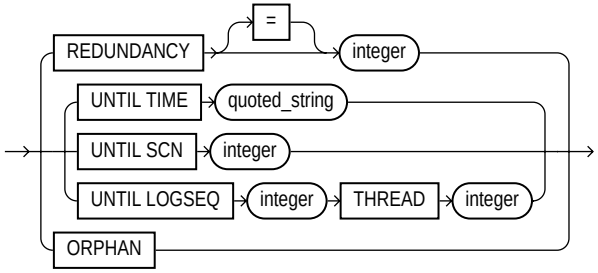


rmanCmd

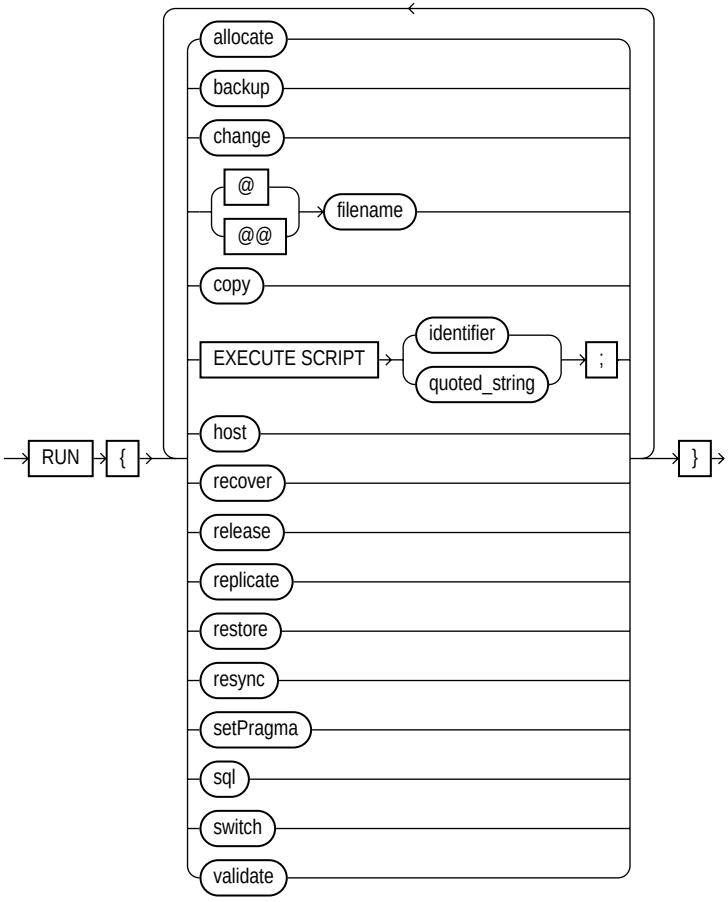

allocateForDelete
@
@@
filename
catalog
change
connect
createScript
deleteScript
EXIT
host
list
printScript
register
releaseForDelete
replaceScript
report
reset
resync
run
set
sql

reportObsoleteOperand

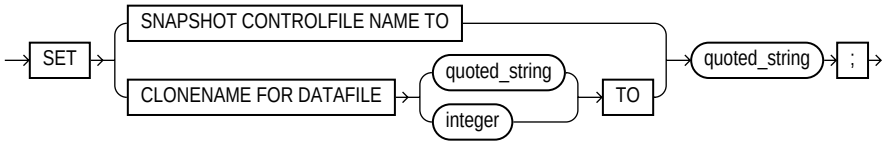
reportObsoleteOperand



run

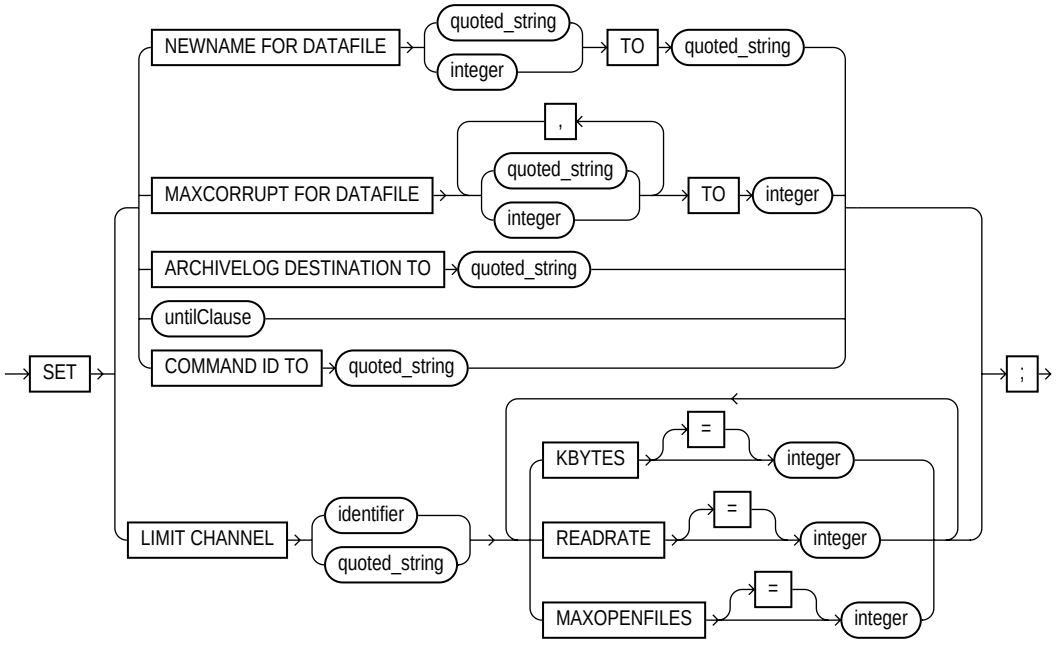

RUN
{
allocate
backup
change
@
@@
filename
copy
EXECUTE SCRIPT
identifier
quoted_string
;
host
recover
release
replicate
restore
resync
setPragma
sql
switch
validate
}

set


SET
SNAPSHOT CONTROLFILE NAME TO
CLONENAME FOR DATAFILE
quoted_string
integer
TO
quoted_string
;

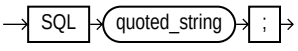
setPragma

setPragma



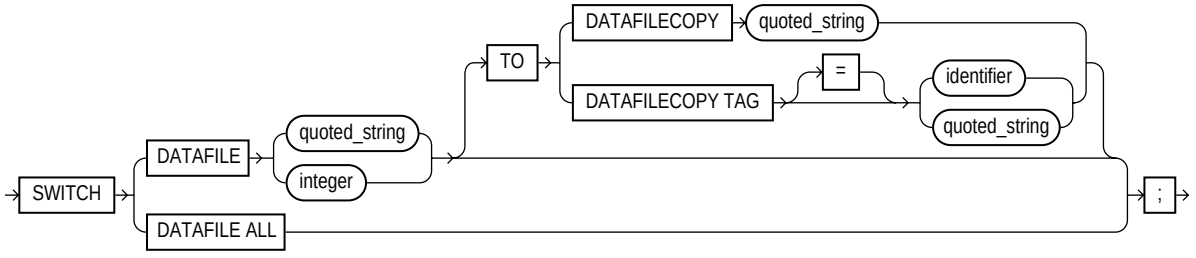
sql

sql



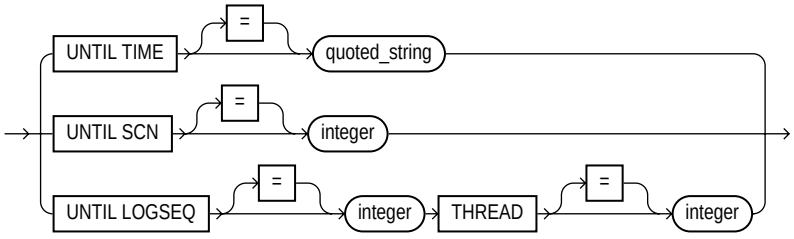
switch

switch



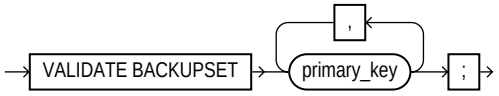
untilClause

untilClause



validate

validate



Glossary

ATL (automated tape library)

A unit which contains one or more tape drives, a robotic arm, and a shelf of tapes. The ATL is able to load and unload tapes into the tape drive from the shelf without operator intervention. More sophisticated tape libraries are able to identify each tape; for example, the robotic arm may have a bar-code reader which allows it to scan each tape's barcode and identify it. Synonymous with tape silo.

auxiliary set

A term used in tablespace point-in-time recovery. The auxiliary set is the set of files not in the recovery set, which must also be restored in the clone database, for the point-in-time recovery set to be successful. These include:

- Backup control file
- System tablespace
- Any datafiles containing rollback segments
- Temporary tablespace (optional).

A small amount of space is required by export for sort operations, and if a copy of the temporary tablespace is not included in the auxiliary set then it will be necessary to provide sort space either by creating a new temporary tablespace after the clone has been started up, or by setting autoextend to ON on the system tablespace files. See also **recovery set.**

backup

A representative copy of data, made by taking a backup. In the context of Oracle, a backup could be made by:

- taking a file copy of the data (via export)

- using Recovery Manager to make a backup (datafile copy or backup set)
- either to disk or to tape using operating system utilities (such as cp, tar, dd, VMS)

To take an image copy of datafiles, control files and other Oracle database structures.

To backup: To make a representative copy of data. If the original data is lost, the backup can be used to reconstruct the lost information.

backup—Recovery Manager command

A keyword recognized by Recovery Manager. This keyword tells recovery Manager that the user wishes to backup data, and the data should be written out in a backup-set format (compare this to copy (2)).

backup sets

A backup set is a backup for (one or more) Oracle files, where the files are multiplexed together. The reason for multiplexing is to give performance benefits by:

- not flooding a particular datafile with reads
- keeping high-speed tape drives streaming

Files in a backup set must be extracted using a restore command. There are two types of backup sets:

- datafile backup set

  Is created when a user backs up any datafiles, or a control file. This type of backup only contains datafile blocks that have been used; unused blocks are omitted (see never used blocks and compression).

- archivelog backup set

  Created when a user backs up archived logs.

A backup set is comprised of one or more physical files, called backup pieces.

backup piece

A backup piece belongs to only one backup set. Typically, a backup set would consist of only one backup piece. The only time Recovery Manager creates more than one backup piece is when the piece size is explicitly limited by the user—this is by specifying the keyword ‘KBYTES’ in a setlimit command. The only reason to specify KBYTES in a backup command, is when the storage or media manager you are writing your backup to is not able to support writing a file larger than ‘KBYTES’ in size.

backup—whole database

A whole backup is a backup of the control file and all datafiles which belong to a database. Whole backups can be taken at any time (i.e. the database can be mounted, open, closed etc.).

block—Oracle

The smallest unit of an Oracle database, the size of which is determined by the parameter DB_BLOCK_SIZE at database creation.

change vector

A single change to a single data block. A change vector is the smallest unit of change. Also see redo record.

channel

A channel is a Recovery Manager resource allocation. Every channel allocated starts a new Oracle server process, which performs backup restore and recovery actions. The more channels allocated the greater the potential degree of parallelism for backup, restore and recovery. The type of channel specified will determine whether the Oracle server process will attempt to read and write to disk, or through the Media Management Interface e.g.

- If the channel is of type disk, the server process will attempt to read backups from, or write backups to disk.
- If the channel is of type ‘SBT_TAPE’, the server process will attempt to read backups from or write backups to a Media Manager.

Channels are always able to read and write datafiles to and from disk, no matter what their type.

checkpoint

When a checkpoint occurs, all modified database buffers in the SGA are written to the datafiles by DBWR. Once the checkpoint completes, the datafiles and control files are modified to reflect the fact that it has successfully completed.

checksum

An Oracle internal computed number stored in an Oracle block, which allows Oracle servers to validate the consistency of the block.

clean shutdown (see shutdown clean)

clone database

A subset of a database which is restored to a new location, and started up with a new instance name. The intended use of a clone database is usually to perform tablespace point-in-time recovery. A clone database consists of the recovery set and auxiliary set. The clone database has various substantive differences from a regular database copy.

closed backup

A backup of one or more database files taken while the database is closed. Typically, closed backups are also whole database backups. If the database was closed cleanly, it means the all the files in the backup are consistent (see database backups - consistent, for more information). If the database was closed using a shutdown abort or the instance terminated abnormally, the backup must be recovered before it is consistent.

closed database

See database—closed

cold backup

See closed backup

complete recovery

See recovery—complete

control file

A file associated with a database that maintains the physical structure and timestamps of all files in that database. The control file is updated continuously during database use, and must be available for writing whenever the database is mounted or open.

control file—backup

A backup control file is a backup of the control file. Backup control files are typically restored when all copies of the current control file are damaged, and are sometimes restored before performing certain kinds of point-in-time recovery.

control file—current

The current control file is the control file on disk - it is the most recently modified control file for that incarnation of the database. For a control file to be considered current during recovery, it must not have been restored from backup.

consistent database backups

See **database backups—consistent**

copy—generic term

To copy:

To make a replica of an original.

It is possible to take copies of Oracle datafiles, control files, and archived redo logs in two ways: 1. using Operating System utilities (on Unix one could use cp, or dd). 2. using the Recovery Manager **copy** command.

copy—Recovery Manager command

A Recovery Manager keyword, which when used, will make a replica of a database's datafiles, control file or archived redo log. This replica is made by Oracle server process, allocated to a Recovery Manager channel, which reads the Oracle file, and writes a replica out to disk. An open database's datafiles can be copied using Recovery Manager without putting the databases' tablespaces into hot backup mode.

corrupt block

An Oracle block which is not in a recognized Oracle format, or whose internal contents are not consistent. Oracle identifies corrupt blocks as one of two types:

- logically corrupt: The block was corrupted by the application of redo
- media corrupt: The block format is not correct. To be identified as media corrupt, the block may:
  - have an impossible format
  - have a wrong data block Address
  - impossible block type

The only way a media corrupt block can be repaired, is:

- by replacing the block, and recovering. For example, the block is replaced when the datafile is restored, or when the block is replaced by a newer version when applying an incremental backup.
- by renewing the block. This happens when the table is dropped, and the blocks are reused for another object

If the media corruption is due to faulty hardware, neither solution above will work until the hardware fault is rectified.

corrupt datafile

A datafile that contains one or more corrupt blocks.

database backup—consistent

A consistent backup is whole database backup (all datafiles and control file) which can be opened with the RESETLOGS option without requiring any recovery (that is, you do not need to apply redo to any files in this backup for it to be consistent). Such a backup would have to be taken after a database has been shutdown cleanly. The database must remain closed until the backup has completed.

All files in a consistent backup (including the control file) must be checkpointed at the same SCN, and must not contain changes past that SCN. The only files in a consistent backup which are allowed to have older SCNs, are files in tablespaces which are read-only or offline normal.

database backup—inconsistent

An inconsistent database backup is a backup whereby some of the files in the backup contain changes which were made after the files have been checkpointed. This type of backup needs recovery before it can be made consistent. This type of backup is usually created by taking open-database backups; that is, the database is open while the files are being backed up. An inconsistent backup can also be created by backing up datafiles while a database is closed, either:

- immediately after an Oracle instance crashed (all instances in an Oracle Parallel Server cluster)
- after shutting down the database using shutdown abort

database—closed

A database which is not available to users to query and update. Although the database is closed:

- the database could be mounted (but not open)
- the instance could be started (but not have the control file mounted)

database—mounted

An instance which is started, and has the control file s associated with the database, mounted (that is, the control file s are open). A database is typically put in a mounted (but not opened) state for maintenance, or for restore and recovery.

database—open

A database which is available to users to query and update. The database is opened either automatically via a startup command or explicitly, via an alter database open command.

datafile

A datafile is a physical Operating System file on disk created by Oracle, which contains data structures such as tables and indexes. A datafile can only belong to one database (and in fact, can only belong to one tablespace).

datafile—inaccessible

A datafile which Oracle is attempting to read, but cannot be found. Attempts to access an inaccessible file results in errors. Typically, a file is inaccessible because the media on which it is stored is faulty, or the file has been moved or deleted.

datafile copy

A copy of a datafile on disk produced by either:

- a Recovery Manager copy command, or
- an Operating System backup of a datafile, which has been cataloged

datafile header

See **file header**

duplexed archived redo log

The Oracle archiver process (ARCH) is able to archive two copies of a redo log. This is called duplexing the archived redo log. This is done by setting LOG_ARCHIVE_DUPLEX_DEST in your init.ora file.

file header

The first few blocks of an Oracle datafile. The file header contains bookkeeping information related to that file.

fractured block

A type of media corruption which could occur when backing up an open database. This phenomenon occurs when DBWR is writing a block, at the same time an Operating System utility is reading the block for backup. The block the Operating System read may be 'split' i.e. the top of the block is written at one point in time, and the bottom of the block is written at another point in time. If this backup were used to restore from, and Oracle were to read the block, the block would be considered to be “corrupt.”

fractured block and Operating System Backups

This possible timing mismatch between Oracle writing a block, and the Operating System reading the same block is the reason tablespaces must be put in hot-backup mode before initiating Operating System backups. A database in hot backup mode writes whole Oracle data blocks out to the redo log, hence if a block is split in the backup, it will be repaired by rolling forward using the redo. Recovery Manager does not suffer from this problem, as the server process performing the backup (backup set or copy) detects if a block it reads for backup is split, and re-reads the block to get a consistent version.

full backup

A Recovery Manager backup set, which is not an incremental. Note that full does not refer to *how much* of the database, full refers to the fact that the backup was not an incremental backup. Hence it is possible to take a full backup of one datafile. Note it is possible to apply incremental backups of levels > 0 on top of a full backup, if it will roll the datafiles forward to a later time. The only difference between a full backup and an incremental level 0, is the full backup will not affect the number of blocks backed up by any subsequent incrementally. Compare this with **backup—whole database**.

full export

An export of the whole database.

full resync

See **resync— full**

hot backup

See **open backup**

hot backup mode

A tablespace is put into “hot backup mode” when the command ALTER TABLESPACE <tablespace name> BEGIN BACKUP is run. This command is executed before taking open-database (that is, hot) backups.

This command is necessary before a user takes an operating system backup of (one or more) datafiles which compose the tablespace being backed up. It is not correct to use hot-backup mode when you use Recovery Manager to take backups. See **fractured block**

inaccessible datafile

See **datafile—inaccessible**

incarnation (database)

This term is used to identify the different “versions” of the same physical database. The incarnation of the database changes once it has been opened with the RESETLOGS option, to be able to distinguish it from the previous “version.”

inconsistent database backups

See **database backups—inconsistent**.

incremental backup

Incremental backups are a method by which you only backup modified blocks. An incremental level 0 backup, performs the same function as a full backup in that they both backup all blocks that have ever been used. The difference between an incremental level 0 backup and a full backup, is that a full backup will not affect what blocks are backed up by subsequent incremental backups, whereas an incremental backup will affect what blocks are copied out by subsequent incremental backups.

Incremental backups of levels > 0 backup only blocks which have changed since previous incremental backups. Blocks which have not changed will not be backed up.

instance

A set of memory structures and Oracle code (including the background processes) resident in memory. The way an instance becomes created and resident in memory is by issuing a startup command. An instance can become resident in memory by issuing any of the following commands:

- STARTUP NOMOUNT—the instance starts, but does not mount the control files nor open the database.
- STARTUP MOUNT—The instance starts, then mounts the database's control file s. It does not open the database. See **database—mounted**.
- STARTUP—The instance starts, mounts the database's control files, and opens the database. See **database—open**. An instance is removed from memory by issuing a shutdown command.

media manager

A utility provided by a third party vendor which is capable of actions such as loading, labelling and unloading sequential media, such as tape drives. Media Managers also allow you to configure media expiration and recycling, and typically have the ability to control Automated Tape Libraries.

media management interface

(also known as:

- the Media Management Layer
- Media Management Library—MML
- SBT Interface)

An Oracle published interface to which Media Management vendors have written compatible software libraries. This software integrates with Oracle so that an Oracle server process is able to issue commands to the Media Manager to write backup files out to sequential storage, and read files from sequential storage. The media manager will handle the actions required to load, label and unload the correct tape, when Oracle issues a request to backup or restore a file.

mirror

A term used to indicate there is more than one exact copy of data. This term is most widely used to indicate that hard disks are mirrored, so that if one of the disks becomes unavailable, the other disk can continue to service requests without interruptions.

mirror—breaking

To break a mirror is to tell the operating system or hardware managing the mirror that you no longer want the mirror image to be kept up-to-date. This is one method used for creating operating system database backups: all the tablespaces in the database are put in hot backup mode and the mirror is broken. All tablespaces are then taken out of hot-backup mode. A backup to tape is then taken from the broken half of the mirror. After the backup is complete, the mirror can then be resilvered.

mirror—resilvering

To resilver a mirror is to tell the operating system or hardware managing the mirror that you want to refresh the old broken mirror from the half that is up-to-date, and then maintain both sides of the mirror.

mirrored control file

See **multiplexed control file**

mirrored online redo log

See **multiplexed online redo log**

multiplexed

- **online logs**

  Oracle is able to maintain more than one copy of the online log automatically: this is called multiplexing the online log. This is controlled by creating multiple members in each redo log group. The degree of multiplexing is directly related to the number of members in each group. Also known as mirrored online redo logs

- **control file**

  Oracle is able to maintain more than one copy of a database's control file; this is called multiplexed control files. This is controlled by specifying multiple entries in the init.ora CONTROL_FILES parameter. Also known as mirrored control files.

- **backup set**

  Datafile blocks included in the same backup set are multiplexed together. This means that blocks from files in this set are interspersed with blocks from the other files in that set.

- **archived redo logs**

  see **duplexed archive redo logs**

never-used blocks and compression

A never-used block is an Oracle block which has never contained any data. An newly created datafile would contain many never-used blocks. When Oracle writes out backup sets, it only includes blocks which have been used (it does not write out never used blocks) i.e. the datafiles are “compressed.”

offline tablespace

A tablespace which is not available to users when the database is open. A tablespace can only be taken offline by a DBA while the database is open. If a tablespace is taken offline, all files that comprise the tablespace (that are not already offline) are taken offline.

A tablespace may be taken offline with three different priorities:

- offline normal (or clean)

  All the files in the tablespace are checkpointed, then made offline. If any file belonging to the tablespace is not available, the tablespace cannot be taken offline normal. Files in a tablespace taken offline cleanly do not need to be recovered before the tablespace is brought back online.

- offline temporary

  All files in the tablespace which are accessible to Oracle are checkpointed, then made offline. Files which were checkpointed by the offline-temporary command do not need recovery, however files which were not checkpointed (because they were not accessible at the time of the offline immediate command) must be recovered before the tablespace is brought back online.

- offline immediate

  All files in the tablespace are taken offline without any attempt to checkpoint the files first. All files in the tablespace must be recovered before the tablespace is brought online. The database must be open to take a tablespace offline.

offline backup (see closed backup)

This term is no longer used to refer to a closed database backup, as it causes confusion with offline tablespaces and datafiles.

offline database (see database—closed)

This term is no longer used to refer to a closed database, as it causes confusion with offline tablespaces and datafiles.

offline datafile

A datafile which is not available to users when the database is open. A datafile can be taken offline either as a part of a tablespace offline or a datafile can be taken offline individually. In exceptional circumstances, Oracle will automatically take a datafile offline if it is required.

A datafile may be taken offline either:

- as a consequence of the tablespace it belongs to being taken offline (see to offline—tablespace) automatically by Oracle, if there is a failure which necessitates this. This file will need recovery before it can be brought online.
- by issuing the command ALTER DATABASE DATAFILE <filename> OFFLINE.

A datafile taken offline by the alter database datafile command must be recovered before being brought back online. The offline command can be executed while the database is mounted or open.

online tablespace

A tablespace which is available to users if the database is open. A tablespace is made available for access by users by issuing the command ALTER TABLESPACE <name> ONLINE. The database must be open to alter a tablespace online, and all files in the tablespace must be consistent with the rest of the database before the tablespace can be made online.

online backup (see open backup)

This term is no longer used to refer to an open-database backup, as it causes confusion with online tablespaces and datafiles.

online database (see database—open)

This term is no longer used to refer to an open database, as it causes confusion with online tablespaces and datafiles.

online datafile

A datafile which is made available for access by users. The database can be open or mounted to issue the command ALTER DATABASE DATAFILE <filename> ONLINE. If the database is open, the file must be consistent with the remaining database before the it can be made online. If the database is mounted, the file can be made online without being consistent with the other datafiles (typically this is done before initiating a database recovery).

open backup (also known as open-database backup, or hot backup)

A backup of one or more datafiles taken while a database is open. This can be an operating system backup (which requires the tablespaces to be in hot backup mode), or a Recovery Manager backup (which does not need the tablespaces to be in hot backup mode).

open database

see database—open

parallelization

Recovery Manager

To parallelize backup and restore using Recovery Manager multiple channels must be allocated.

Backup Set creation

Is parallelized by allocating multiple channels and also specifying **filesperset**.

File Copy creation

Is parallelized by allocating multiple channels, and by including multiple files to be copied within a single copy command.

Restore Parallelization

Recovery Manager parallelizes restore. The degree of parallelism depends on the number of channels allocated, and also the distinct number of backup sets or file copies which are available to be read from.

Recovery Parallelization

Recovery Manager also parallelizes recovery when it is applying incremental backups. The degree of parallelism depends on the number of channels allocated, and also the distinct number of backup sets which are available to read from.

Recovery Parallelism

Irrespective of whether you use Recovery Manager or O/S backups, it is possible to use recovery_parallelism to parallelize the application of redo logs.

partial resync

See **resync—partial**

password files

A file created by orapwd command.

A database must use password files if you wish to connect internal over a network. For a more comprehensive explanation, see <Title>Oracle8 Server Administrator's Guide

read-only tablespace

See t**ablespace— read onl**y

recover

To make a restored file current.

or

To make a restored file current to a specific point in time. In ARCHIVELOG mode, you have the choice of complete or incomplete recovery (see **recovery—complete**, and **recovery —incomplete**).

For NOARCHIVELOG mode databases, the only option is to restore from the most recent backup. In exceptional circumstances, it may be possible to recover a datafile or database if the database is not in ARCHIVELOG mode. The only time it is possible to recover a NOARCHIVELOG mode database, is if none of the online logs have been overwritten since the backup

For example:

1. The DBA makes a closed (consistent) backup of a database with 10 online log groups
2. The DBA opens the database and 5 log switches occur (that is, less than or equal to 9 logs can be switched with 10 online logs)
3. Media failure occurs which damages one or more datafiles

In this case all redo generated since the backup is still available, which means that recovery is possible.

Note that this situation is *not* typical. If you must be able to recover the database—not just restore it—the database must be in ARCHIVELOG mode. Note that you should not be backing up your online log, so this is *not* an option if you need to be able to recover your database in case of media failure.

recover—Recovery Manager

A Recovery Manager command which will make a restored datafile newer by the application of incremental backups (if they exist) and then by the application of redo (archived and online redo logs).

RECOVER—SQL command

A SQL command issued to make a restored file newer by the application of redo (archived and online redo logs).

recovery—complete

To restore and recover a database and apply all redo generated (both in archived and online redo logs) since the backup. This type of recovery is typically performed when media failure damages one or more datafiles, or control file s. The damaged files are fully recovered using all redo generated since the restored backup was first made.

recovery—incomplete

To restore a database from backup, and recover the database, but not apply all of the redo information generated since the backup.

Incomplete recovery is usually performed when:

- The online logs are lost due to hardware failure

  In this case the database is recovered until the last archived log generated before the failure.

- A user error necessitates recovery up until just before the error occurred

  The requirement is to recover up until some point in time before an incorrect action occurred in the database. For example, a user mistakenly deletes payroll transactions before the transactions are sent to the payroll agency. In this example, the DBA will need to restore the whole database and then perform incomplete recovery up until the point just before the user deleted the transactions.

- An archived log required for recovery is missing

  An archive log which is needed for complete recovery was not backed up, or the archived log contents are corrupt (e.g. due to media failure). In this case, the only option is to recover up to the missing log.

  In all cases above, the database must be opened with the RESETLOGS option.

recovery catalog

A recovery catalog is a set of Oracle tables and views which are used by Recovery Manager to store information about Oracle databases. This information is used by Recovery Manager to manage the backup, restoration and recovery of Oracle databases.

recovery catalog database

An Oracle database which contains a recovery catalog schema.

Recovery Manager

A tool used by DBAs to backup, restore and recover Oracle databases (datafiles, control files and archived logs). It can be used with a Recovery Catalog, or without. If a recovery catalog is not used, Recovery Manager decides how to backup, restore and recover the database using the database's control file.

recovery set

A subset of a database's tablespaces that require point-in-time recovery to be performed on them. Also see auxiliary set.

recovery—point-in-time

database

A recovery of all datafiles and control file to a specific time which is not a full recovery.

tablespace

A recovery of all datafiles in a tablespace to a specific time which is not the full recovery. This recovery is done in a clone database. The tablespace is then re-integrated into the original database. This feature is new to Oracle8.

redo log

online redo log

The online redo log is a set of two or more files which record all changes made to Oracle datafiles and control files. They are written to by LGWR. See also **multiplexed online redo log**.

current online redo log

The online redo log file (or group, in the case of multiplexed logs) which is currently being written to by LGWR.

archived redo log

This is also called the offline redo log. If the database is in ARCHIVELOG mode, as each online redo log is filled, it is copied to one (or more) archive log destination(s). This copy is the archived redo log.

See also **duplexed archived redo logs**

redo log groups

Each online redo log belongs to a group. A group must have at least one member, but may have more. If a redo log group has more than one member, it is said to be multiplexed.

thread

Each Oracle instance has its own set of online redo log groups. These online redo log groups are called an instance's thread of online redo. In non-Oracle Parallel Server environments, one database has only one thread, belonging to the instance accessing it. In Oracle Parallel Server environments, each instance has a separate thread (i.e. each instance has it's own online redo log). Each thread has its own current log member.

redo record

A redo record is a group of change vectors describing a single, atomic change to the database. Redo records are constructed for all data block changes, and saved on disk in the online redo log. Redo records allow multiple database blocks to be changed so that either all changes occur, or no changes occur, despite arbitrary failures.

register database

A Recovery Manager command which registers a database with the recovery catalog. Any resync catalog commands will result in the recovery catalog being updated with the database's current configuration.

RESETLOGS option

A method for opening a database which results in the database incarnation changing, the log sequence number being reset to 1, and the online logs being reformatted (or created, if they do not already exist). Typically, a database is opened with the RESETLOGS keyword:

- after incomplete recovery (until time, log sequence, change, cancel)
- after recovery using a backup control file

The reason for opening a database with this option is to prevent database corruption by ensuring that any archived logs created by that database in the past cannot erroneously be applied to this incarnation of the database.

restore

To reconstruct or bring back an original copy of a file from backup.

resync—full

A full catalog resync is a Recovery Manager operation which refreshes the Recovery Catalog with all changed information in the database's control file. Full resyncs should be initiated by a DBA frequently, by issuing the Recovery Manager command **resync catalog**. Full resyncs can also be initiated by Recovery Manager if it determines this action is necessary before executing certain commands.

resync—partial

Partial resyncs transfer information to the Recovery Catalog about archived redo logs, backup sets and datafile copies. Partial resyncs will not transfer information such as:

- new datafiles
- new or removed tablespaces
- new or removed online log groups and members

Partial resyncs are initiated by Recovery Manager:

- before: backup, copy, restore, recover, list and report, if

Recovery Manager determines that a resync is necessary

- after: backup, copy, restore, switch, register, reset and catalog

roll forward

Application of redo logs to datafiles and control files in order to recover them (that is, to recover changes to those files).

SBT

Acronym for System Backup to Tape

See **Media Management Interface**

shutdown clean

A database which has been shutdown with the immediate, transactional or normal options. A database shutdown cleanly does not require recovery; it is already in a consistent state.

snapshot control file

A copy of a database's control file taken by Recovery Manager. The snapshot control file is used by Recovery Manager to read a consistent version of a control file (either to resync the recovery catalog, or if a recovery catalog is not used, to be read by Recovery Manager to decide what actions to perform). A snapshot control file is created by Recovery Manager using the same Oracle code which creates backup control files (ALTER DATABASE BACKUP CONTROL FILE TO '<location>').

split block (see fractured block)

staging (of archived redo logs from tape to disk)

Staging of archived logs is a term used to describe restoring archived logs from tertiary storage (usually tape) to disk, in order to allow recovery to proceed.

switch

A Recovery Manager command which converts a datafile copy into a datafile used by an Oracle database. It performs the equivalent function of the SQL command ALTER DATABASE RENAME DATAFILE FROM '<original name>' TO '<new name>', and also marks the datafile copy as no longer available.

tablespace

A database is divided into one or more logical storage units called tablespaces. Each tablespace has a number of datafiles exclusively associated with it.

tablespace—read only

A tablespace whose status has been changed to prevent it from being updated. It is put in read-only mode by executing the SQL statement ALTER TABLESPACE <tablespace> READ ONLY. Typically, the reason for putting a tablespace in read-only mode is to be able to reduce the frequency it is backed up. For example, instead of backing up the tablespace nightly, it may be feasible to reduce the backup frequency, say to once a month, or less.

Note: The longer the duration between backing up any tablespace, the longer you will need to retain your backup media and the larger the risk of failed backup media (as you will have copied it out fewer times).

---

tablespace point-in-time recovery

See **recovery—point-in-time tablespace**

tape streaming

To write output to a tape drive fast enough to keep the tape constantly busy.

tape drive

A piece of hardware which reads and writes magnetic tapes.

tape volume

A term used by many vendors to mean one physical piece of tape media.

Index

A

ABORT option
  SHUTDOWN command, 12-10, 12-22, 12-26, 12-52
ACTIVATE STANDBY DATABASE option
  ALTER DATABASE command, 12-38
active online redo log
  loss of group, 12-50
ALERT file, 2-18, 12-3, 12-25, 12-29, 12-34
allocate channel command, 8-10, 8-28, 8-36
  type disk operand, 8-10
allocate channel for delete command, 8-7, 9-8
ALTER DATABASE command
  ACTIVATE STANDBY DATABASE option, 12-38
  BACKUP CONTROLFILE option, 2-23
  BACKUP CONTROLFILE TO TRACE option, 11-10, 12-53
  CLEAR LOGFILE GROUP option, 12-37
  CLEAR LOGFILE option, 12-48
  CREATE DATAFILE option, 12-13, 12-39
  CREATE STANDBY CONTROLFILE option, 12-36
  DATAFILE ONLINE option, 12-28, 12-32
  MOUNT STANDBY DATABASE option, 12-37, 12-43
  NORESETLOGS option, 11-11, 12-25, 12-29, 12-34, 12-53
  OPEN NORESETLOGS option, 12-17
  OPEN option, 12-52
  OPEN RESETLOGS option, 8-4, 12-10, 12-25, 12-29, 12-34
  RECOVER AUTOMATIC LOGFILE option, 12-7
  RECOVER clause, 12-4
  RECOVER LOGFILE option, 12-6
  RECOVER...FROM parameter, 12-5
  RESETLOGS option, 11-11, 12-3
ALTER SYSTEM command
  ARCHIVE LOG CURRENT option, 12-36, 12-38
ALTER TABLESPACE command
  BEGIN BACKUP option
    marking the beginning, 11-6
    minimizing time between begin and end, 11-8
  BEGIN/END BACKUP option, 8-8
  END BACKUP option
    marking the end, 11-6
    minimizing time between begin and end, 11-8
  OFFLINE option, 11-9
  ONLINE option
    bringing tablespaces back online, 11-9
Apply Recovery Archives dialog box, 12-23
  Enterprise Manager, 12-3
ARCHIVE LOG CURRENT option
  ALTER SYSTEM command, 12-36, 12-38
archived logs
  backing up, 9-3
archived redo log, 1-7, 2-13 to 2-15
  applying during recovery, 12-4, 12-6
  deleting after recovery, 12-14
  duplexing, 2-15
  errors during recovery, 12-8
  location during recovery, 12-4
  loss of, 12-51
  preserving log sequence number, 12-24, 12-28,

12-33
resetting log sequence number, 12-24, 12-28, 12-33
restoring to disk, 12-14
time-based recovery, 12-51
tracking, 7-27
archivelog
copies, listing, 8-12
deleting, 8-6
deleting record of, 8-6
marking as available, 8-7
marking as unavailable, 8-7
registering, 8-2
ARCHIVELOG mode
backup options, 3-5
datafile loss in, 12-46
definition, 2-16
distributed database backups, 3-6
overview, 1-7
strategies for backups in, 3-8
archiver process (ARCH), 2-17 to 2-18
archiving online redo log files, 2-14
archiving
automatic, 2-17
manual, 2-18
modes
database, 2-16 to 2-19
automatic archiving, 2-17
auxiliary sets, 10-3
naming datafiles in tablespaces, 10-10
available operand
change command, 8-7

B

backup
after structural changes to database, 3-3
after using RESETLOGS option, 3-5
archived log, 9-3
ARCHIVELOG mode in, 3-8
backup sets, 7-10
checking datafile backup status, 11-7
choosing method, 5-1 to 5-7
consistent whole database, 2-22
constraints, 7-25
control file, 2-24, 9-3, 11-10
creating a strategy, 3-7
cumulative incremental, 7-13, 7-16
database, 9-2
datafile, 2-24
DB_VERIFY utility, 11-4
definition, 1-2, 4-4
distributed databases, 3-6
Enterprise Backup Utility used for, 5-3
Export utility and, 2-28, 3-6
Export utility used for, 5-3
frequency, 3-2
full, 7-12
generating reports for, 8-12
guidelines for, 3-2
image copies, 7-10, 7-17
importance of, 1-2
inconsistent whole database, 2-23
incremental, 7-12, 9-6
definition, 4-4
Recovery Manager, 8-22
individual datafiles, 9-3
keeping, 3-4, 9-7
listing files needed, 11-2
logical
definition, 2-28
marking end, 11-6
NOARCHIVELOG mode, in, 3-7, 9-2
noncumulative incremental, 7-14
of online redo logs, 3-9
of recovery catalog, 6-6
operating system, 5-2
Parallel Server Environment, 9-5
planning before database creation, 3-2
privileges for control files, 11-10
procedures for, 11-2 to 11-14
offline datafiles, 11-8
offline tablespaces, 11-8
online datafiles, 11-5
online tablespaces, 11-5
Recovery Manager types, 7-10
Recovery Manager, user tags for, 7-24
restoring whole database backup, 12-9
scenario, Recovery Manager, 8-27
specification list, 8-24

specifying device, 8-10
tablespace, 3-3, 9-2, 9-3, 11-8
test strategies, 3-7
to disk, 5-4
to disk, using Recovery Manager, 8-10
to sequential media, 5-5
types of, 2-21 to 2-25, 11-2 to 11-14
using stored scripts, 8-19
whole database, 2-21, 11-3 to 11-4
Recovery Manager, 8-22
backup command, 8-21
format operand, 8-20
setmaxcorrupt clause, 9-7
skip inaccessible option, 9-7
skip offline option, 9-2, 9-7
skip read only option, 9-2
tag operand, 8-23
backup constraints, 7-25
backup control files
snapshot, 7-9
BACKUP CONTROLFILE option
ALTER DATABASE command, 2-23
BACKUP CONTROLFILE TO option
ALTER DATABASE command, 11-2
BACKUP CONTROLFILE TO TRACE option
ALTER DATABASE command, 11-2, 11-10, 12-53
backup file
user-created
cataloging, 8-8
backup methods, 5-1 to 5-7
backup object list, 8-24
backup piece
deleting, 8-6
deleting record of, 8-6
marking as available, 8-7
marking as unavailable, 8-7
naming, 8-20
backup pieces, 7-11
backup sets, 7-10
backup pieces, 7-11
compression, 7-11
deleting multi-piece, 9-8
full, 7-12
incremental, 7-12
limiting number of files, 8-20
listing, 8-12, 8-17, 8-18
multiplexed, 7-22
organizing, 8-21
backup specification list, 8-24
Begin Online Tablespace Backup dialog, 11-6
BEGIN/END option
ALTER TABLESPACE command, 8-8
bitmap indexes
and TSPITR, 10-5, 13-5

C

cancel-based recovery, 4-7
procedures, 12-22
catalog command, 8-2, 8-8
catrman.sql script, 6-5
catrman.sql script, 8-3
CE CodeExInd, xvii
change command, 8-6
available operand, 8-7
delete operand, 8-7
unavailable operand, 8-7
uncatalog operand, 8-7
change-based recovery, 12-31 to 12-35
coordinated in distributed database system, 12-2
changed-based recovery, 4-7
change...delete command, 8-8
channel control, 7-19, 8-9
allocating channel, 8-9
channels
allocating, 7-19
checkpoint, 2-5 to 2-9
after a time interval, 2-6
at log switches, 2-6
checkpoint process (CKPT), 2-8
control files and, 2-20
database, 2-6
datafile, 2-6
definnition, 4-4
events during, 2-8
fast, 2-8
forcing, 2-6
incremental, 2-7

normal, 2-8
online redo log files and, 2-5
overview, 2-5
performance effect of, 2-5
shutting down an instance and, 2-6
taking a tablespace offline and, 2-6
types of, 2-6
CLEAR LOGFILE GROUP option
ALTER DATABASE command, 12-37
CLEAR LOGFILE option
ALTER DATABASE command, 12-48
clone databases, 10-3
converted filenames, 10-13
exporting, 13-15
opening, 13-15
preparing for TSPITR, 13-13
preparing parameter files for, 13-12
recovering, 13-14
using datafile copy for TSPITR, 10-11
closed database
recovery, 12-3
cold backups
whole database backups, 11-3
command file for Recovery Manager, 6-10
commands
backup, 7-3
categories of, 7-3
LIST, 7-23
recovery catelog maintenance, 7-3
REPORT, 7-23
REPORT DELETABLE, 7-23
REPORT NEED BACKUP, 7-23
REPORT UNRECOVERABLE, 7-23
restore, 7-3
stored script maintenance, 7-3
commands, Recovery Manager
allocate channel, 8-10, 8-28, 8-36
allocate channel for delete, 9-8
backup, 8-21
catalog, 8-2, 8-8
change, 8-6, 8-7
change...delete, 8-8
copy, 2-27, 8-28
delete script, 8-19
execute script, 8-19
list, 8-12, 9-11
list backupset, 8-18
list copy, 8-18
list incarnation of database, 8-4
print script, 8-19
register database, 8-3
release channel, 8-11
replace script, 8-19
replicate, 8-31
report, 9-12
report deletable, 8-15
report need backup, 8-13
report need backup days, 8-14
report schema, 8-16
report unrecoverable, 8-14
reset database, 8-4
reset database to incarnation, 8-4
restore, 9-10
resync catalog from backup controlfile, 8-9
set newname, 8-32, 8-35
setlimit channel, 8-11
switch, 8-35
committing transactions
writing redo log buffer and, 2-2
COMPATIBLE initialization parameter, 12-41
complete export, 11-14
complete recovery
procedures for, 12-15
consistent whole database backups, 2-22
constraints
backup, 7-25
restore, 7-26
control file, 1-4, 2-19 to 2-20
archived redo log information in, 2-15
backing up, 9-3, 11-2, 11-10
changes recorded, 2-20
checkpoint information, 2-20
contents, 1-4, 2-19
CONTROL_FILES parameter
for primary and standby databases, 12-40
definition, 1-4, 4-3
during incomplete recovery, 12-22, 12-31
during time-based recovery, 12-26
finding filenames, 11-2
inconsistency with data dictionary, 12-25, 12-30

log sequence numbers and, 2-5
loss of, 12-51, 12-52
media recovery and, 2-15
mirrored, loss of, 12-52
multiplexed, 2-20
privileges to backup, 11-10
purpose, 1-4
restore destination, 8-31
standby database, 12-36
control file backup
definition, 2-24
CONTROL_FILES initialization parameter, 8-31, 12-52
coordinated, time-based, distributed database recovery, 12-2
copy command, 8-28
corrupt datafile blocks
records in control file, 8-21
setting maximum for backup, 9-7
corruption detection, 7-18
CREATE CONTROLFILE command
DATAFILE clause, 12-53
standby database, effect on, 12-40
CREATE DATAFILE option
ALTER DATABASE command, 12-13, 12-39
CREATE STANDBY CONTROLFILE option
ALTER DATABASE command, 12-36
creating
standby database, 12-36
cumulative export, 11-14
cumulative incremental backup, 7-13

D

data dictionary
inconsistencies with control file, 12-25, 12-30
views, 11-5
database
altering physical structure
standby database, impact on, 12-39
archiving modes, 2-16 to 2-19
backing up, 9-2
using Recovery Manager, 8-20
closed
recovery, 12-3
db identifier, 8-3
disaster recovery planning, 12-35 to 12-44
files
listing for backup, 11-2
incarnations
listing, 8-13
incremental backup
Recovery Manager, 8-22
modes
NOARCHIVELOG, 2-16
physical structure, 1-4 to 1-7, ?? to 4-5, ?? to 6-13
point-in-time recovery, 9-10
recovery
after control file damage, 12-52
procedures, 12-1 to 12-54
recovery after using RESETLOGS option, 3-10
registering in recovery catalog, 8-2, 8-3
resynchronizing with recovery catalog, 8-4
shutdown for backups, 9-2
standby, 12-35 to 12-44
activating, 12-38
creating, 12-35
maintaining, 12-36
startup
recovery during, 12-4
whole database backup, 11-3 to 11-4
Recovery Manager, 8-22
database buffers
checkpoints and, 2-5
database option
list command, 8-17
DATABASE parameter
RECOVER command, 12-3
database structures
archived redo logs, 1-7
control files, 1-4
datafiles, 1-5
online redo log, 1-5
physical, 1-4 to 1-7, ?? to 4-5, ?? to 6-13
redo log files, 1-5
rollback segment, 1-6
database writer process (DBWR)
checkpoints and, 2-5
datafile, 1-5
adding to primary database

effect on standby database, 12-39
backing up, 2-24, 2-27, 9-3
offline, 11-8
online, 11-5
using Recovery Manager, 8-20
backup needed, listing, 8-14, 8-15
backup sets
listing, 8-17
backups, listing, 8-12
checking backup status, 11-7
converting name for standby use, 12-37
copies to be deleted, listing, 8-15
copies, listing, 8-12, 8-18
copying, 2-27
advantages, 8-28
to disk, 9-5
deleting copy of, 8-6
deleting record of, 8-6
listing, 8-16
for backup, 11-2
loss of, 12-45
marking copy as available, 8-7
marking copy as unavailable, 8-7
missing after recovery, 12-25, 12-30
named in control files, 2-19
overview, 1-5
recovery
guidelines, 8-36
without backup, 12-13
recovery creation, 12-13
registering, 8-2
renaming
effect on standby database, 12-39
restore destination, 8-31
restoring, 8-30
guidelines, 8-32
when database is open, 9-8
switching, 8-35
unrecoverable, listing, 8-14
usage, 1-6
viewing
backup status, 11-7
files needing recovery, 12-12
DATAFILE clause
CREATE CONTROLFILE command, 12-53
DATAFILE ONLINE option
ALTER DATABASE command, 12-28, 12-32
datafile option
list command, 8-17
DATAFILE parameter
RECOVER command, 12-4
days option
report command, 8-15
db identifier, 8-3
DB_BLOCK_MAX_DIRTY_TARGET, 2-7
DB_FILE_STANDBY_NAME_CONVERT
initialization parameter, 12-37
for primary and standby databases, 12-40
DB_NAME initialization parameter, 2-20
DB_VERIFY utility, 11-4
DBA_DATA_FILES view, 11-5
delete operand
change command, 8-7
delete script command, 8-19
deleting
file records using Recovery Manager, 8-7
files using Recovery Manager, 8-7
recovery catalog record, 8-7
stored scripts, 8-19
device_type_list option
list backupset command, 8-18
report command, 8-13
devices
backup, finding, 9-2
disaster recovery
planning for, 12-35 to 12-44
disk failure, 1-4
distributed database
change-based recovery, 12-2
coordinated time-based recovery, 12-2
media recovery and snapshots, 12-3
recovery in, 12-2
taking backups, 3-6
distributed database system
recovery, 12-2
duplexing
archived redo log, 2-15
duplexing archived redo log, 2-15

E

End Online Tablespace Backup dialog, 11-6
Enterprise Backup Utility, 5-3
Enterprise Manager
  Apply Recovery Archives dialog box, 12-3, 12-23
  applying log files, 12-6
  RECOVER DATABASE UNTIL TIME statement, 12-28
  SET AUTORECOVERY ON statement, 12-6
  Shutdown Database dialog box, 12-22, 12-26, 12-52
    Shutdown Abort mode, 12-10
  STARTUP command
    RECOVER option, 12-4
  Startup Database dialog box, 12-27
Enterprise Manager Startup dialog box, 12-23
environment variables
  NLS_DATE_FORMAT, 6-11
  NLS_LANG, 6-11
errors
  archiver process, 2-18
execute script command, 8-19
Export utility, 2-28, 5-3
  backups and, 2-28, 3-6
  exports, types of, 11-14
  read consistency and, 11-13
  using for backup, 11-13
exports
  complete, 11-14
  cumulative, 11-14
  incremental, 11-14
  modes, 11-13

F

failure
  archiving redo log files, 2-18
  disk, 1-4
  during checkpoints, 2-8
  hardware, 1-3
  instance, 1-3
  media, 1-4, 12-45
    multiplexed online redo logs and, 2-9
  process, 1-3
  statement, 1-3
  system, 1-3
  user error, 1-3
file recovery, 1-5
file_name_pattern option
  list command, 8-18
filenames
  listing for backup, 11-2
files
  marking as unavailable, 8-7
format operand
  backup command, 8-20
fractured block detection, 7-26
from backup operand
  restore command, 8-33
from copy operand
  restore command, 8-33
from/until time option
  list command, 8-18
full backup, 7-12

G

group, redo log
  archived redo log, 12-47
  online redo log, 12-47

H

hardware failure, 1-3

I

I/O device
  releasing in Recovery Manager, 8-11
I/O errors
  effect on backup, 8-20
image copies, 7-17
  cataloging, 7-27
  user created, 7-18
Import utility, 11-13
  database recovery and, 11-14
  procedure for using, 11-14
inactive online redo log

loss of, 12-48
incomplete recovery
change-based, 12-31 to 12-35
procedures for, 12-21 to 12-35
time-based, 12-26 to 12-30
inconsistent whole database backups
definition, 2-23
incremental backup, 7-12, 9-6
Recovery Manager, 8-22
incremental export, 11-14
incremental option
report command, 8-14
initialization parameters
COMPATIBLE, 12-41
CONTROL_FILES, 8-31, 12-52
for primary and standby databases, 12-40
DB_FILE_STANDBY_NAME_CONVERT, 12-37
for primary and standby databases, 12-40
DB_NAME, 2-20
for primary and standby databases, 12-40
LOG_ARCHIVE_DEST, 12-5, 12-23, 12-28
LOG_ARCHIVE_FORMAT, 12-5
LOG_ARCHIVE_START, 2-18
LOG_CHECKPOINT_INTERVAL, 2-6
LOG_CHECKPOINT_TIMEOUT, 2-6
LOG_CHECKPOINTS_TO_ALERT, 2-9
LOG_FILE_STANDBY_NAME_CONVERT, 12-37
PARALLEL_MAX_SERVERS, 12-11
RECOVERY_PARALLELISM, 12-11
init.ora file, 8-35
installing
recovery catalog, 8-2
instance
failures in, 1-3
recovery during startup, 12-4
integrity checking, 7-26
interrupting media recovery, 12-9

L

list backupset command, 8-18
device_type_list option, 8-18
LIST command, 7-23
list command, 8-12, 9-11
database option, 8-17
datafile option, 8-17
file_name_pattern option, 8-18
from/until time option, 8-18
list_object_list option, 8-17
list_operand option, 8-18
list_qualifier option, 8-18
redundancy option, 8-15, 8-16
tablespace option, 8-17
tag option, 8-18
list copy command, 8-18
list incarnation of database command, 8-4
list_object_list option
list command, 8-17
list_operand option
list command, 8-17, 8-18
list_qualifier option
list command, 8-18
log files
converting name for standby use, 12-37
log sequence number
control files and, 2-5
multiplexed redo logs and, 2-10
preserving after recovery, 12-24, 12-25, 12-28, 12-29, 12-33
requested during recovery, 12-4
resetting after recovery, 12-25, 12-29
resetting to 1, 12-24, 12-28, 12-33
log switch record
recovery catalog, 8-6
log switches
description, 2-5
log sequence numbers, 2-5
mirrored redo log files and, 2-10
log writer process (LGWR)
archiver process (ARCH) and, 2-14
manual archiving and, 2-18
multiplexed redo log files and, 2-10
online redo logs available for use, 2-3
trace files and, 2-11
writing to online redo log files, 2-2, 2-3
LOG_ARCHIVE_DEST initialization parameter, 12-5, 12-23, 12-28
LOG_ARCHIVE_FORMAT initialization parameter, 12-5

LOG_ARCHIVE_START initialization parameter, 2-18
LOG_CHECKPOINT_INTERVAL parameter, 2-6
LOG_CHECKPOINT_TIMEOUT parameter, 2-6
LOG_CHECKPOINTS_TO_ALERT initialization parameter, 2-9
LOG_FILE_STANDBY_NAME_CONVERT initialization parameter, 12-37
logical backup
  definition, 2-28
LOGSOURCE parameter
  SET command, 12-5
loss of inactive log group, 12-48

M

manual archiving, 2-18
media failure, 1-4
  archived redo log file loss, 12-51
  complete recovery procedures, 12-15 to 12-21
  control file loss, 12-51, 12-52
  datafile loss, 12-45
  description, 12-45
  NOARCHIVELOG mode, 12-9
  online redo log group loss, 12-47
  online redo log loss, 12-46
  recovery
    distributed databases, 12-2
  recovery procedures
    examples, 12-45 to 12-54
Media Management Library (MML), 5-6
media recovery, 4-7
  applying archived redo logs, 12-4
  at instance startup, 12-4
  cancel-based, 12-21
  change-based, 12-21, 12-31 to 12-35
  commands, 12-11 to 12-12
  completion of, 12-17, 12-19, 12-21
  deciding which files need recovery, 12-12
  distributed database
    coordinated time-based, 12-2
  error messages, 12-8
  errors with redo log files, 12-8
  from NOARCHIVELOG mode, 12-9
  interrupting, 12-9
  lost files
    lost archived redo log files, 12-51
    lost control files, 12-51
    lost datafiles, 12-45
  lost mirrored control files, 12-52
  NOARCHIVELOG mode, 12-9
  online redo log files, 12-46
  open database-offline tablespace, 12-17
  preparing for, 12-11 to 12-15
  procedures for complete, 12-15 to 12-21
  procedures for incomplete, 12-21 to 12-35
  requirements of incomplete, 12-24, 12-28, 12-33
  restoring
    archived redo log files, 12-14
    damaged files, 12-13
    whole database backups, 12-9
  resuming after interruption, 12-9
  snapshots and, 12-3
  starting, 12-15
  successfully applied redo logs, 12-8
  SYSTEM tablespace, 12-17
  time-based, 12-21
  time-based incomplete, 12-26 to 12-30
  undamaged tablespaces online, 12-17
  using Recovery Manager, 8-35
mirrored control file
  loss of, 12-52
mirrored online redo log, 2-10
  loss of, 12-47
  loss of member, 12-47
mode
  ARCHIVELOG, 1-7, 2-16
  NOARCHIVELOG, 1-7, 2-16
    recovery from failure, 12-9
MOUNT option
  STARTUP command, 12-23, 12-27
MOUNT STANDBY DATABASE option
  ALTER DATABASE command, 12-37, 12-43
multiplexed
  control files, 2-20
  redo log files, 2-9
multiplexed backup sets, 7-22

N

NLS_DATE_FORMAT environment
variable, 6-11, 9-4, 9-10
NLS_LANG environment variable, 6-11, 9-4, 9-10
NOARCHIVELOG mode
backing up, 9-2
backup options, 3-5
datafile loss in, 12-45
definition, 2-16
disadvantages, 12-9
distributed database backups, 3-6
overview, 1-7
recovery from, 12-9
strategies for backups in, 3-7
noncumulative incremental backup, 7-14
nonpartitioned global indexes
and TSPITR, 10-5, 13-5
NORESETLOGS option, 12-24 to 12-25, ?? to 12-26
ALTER DATABASE COMMAND
backing up control file, 11-11
ALTER DATABASE command, 12-53

O

object list
recover command, 8-38
objects owned by SYS
and TSPITR, 10-4, 13-4
offline backups, 3-2
OFFLINE option
ALTER TABLESPACE command, 11-9
online backups, 3-2
ONLINE option
ALTER TABLESPACE command, 11-9
online redo log, 1-5, 2-2 to 2-13, 12-48
active group, 12-47
applying during recovery, 12-4
archived group, 12-47
current group, 12-47
inactive group, 12-47
listing log files for backup, 11-2
loss of
recovery, 12-46 to 12-50
loss of active group, 12-50
loss of all members, 12-47
loss of group, 12-47
loss of mirrored members, 12-47
multiple group loss, 12-50
multiplexed, 1-5
overview, 1-5
preserving log sequence number, 12-24, 12-33
resetting
effect of, 12-24 to 12-25
procedures, 12-24 to 12-26
resetting log sequence number, 12-24, 12-33
status of members, 12-47
online redo logs
backing up, 3-9
clearing, 12-37
OPEN DATABASE
RESETLOGS option, 3-10
open database backup
fractured block detection during, 7-26
OPEN option
ALTER DATABASE command, 12-52
OPEN RESETLOGS option
ALTER DATABASE command, 8-4, 12-10,
12-25, 12-29, 12-34
operating system backup, 5-2
ORA-01578 error message, 12-43
Oracle Enterprise Manager, 5-3
Oracle8 utilities
Recovery Manager, 7-2
OS utilities, 9-7

P

PARALLEL clause
RECOVER command, 12-11
parallel recovery, 12-11
Parallel Server
backups and, 9-5
threads of online redo log, 2-13
PARALLEL_MAX_SERVERS initialization
parameter, 12-11
parallelization, 7-20
factors affecting degree of, 7-20
partitioned tables
and dropped partitions, 13-21

and split partitions, 13-25
and TSPITR, 10-5, 13-4
performing partial TSPITR, 13-16
password file
connecting to Recovery Manager with, 6-8
connecting to Recovery Manager without, 6-7
using with Recovery Manager, 6-6
performance
checkpoint effect, 2-5
physical database structures, 1-4 to 1-7, ?? to 4-5, ?? to 6-13
point-in-time recovery
database, 4-6, 9-10
tablespace, 4-6
preface
Send Us Your Comments, xiii
primary database
preparing for use, 13-16
primary databases
preparing for TSPITR, 13-12
print script command, 8-19
privileges
backing up
control files, 11-10
process failure, 1-3

Q

querying
recovery catalog, 9-12

R

read consistency
Export utility and, 11-13
read-only tablespaces
backing up, 9-2
effect on recovery, 12-44
recovering, 12-44
RECOVER AUTOMATIC LOGFILE option
ALTER DATABASE command, 12-7
RECOVER clause
ALTER DATABASE, 12-4
RECOVER command
DATABASE parameter, 12-3
DATAFILE parameter, 12-4
PARALLEL clause, 12-11
TABLESPACE parameter, 12-4
unrecoverable objects and standby databases, 12-43
recover command, 8-35, 8-38
Recovery Manager, 4-2
RECOVER DATABASE command, 12-52
RECOVER DATABASE UNTIL CANCEL statement, 12-23
RECOVER DATABASE UNTIL CHANGE statement, 12-32
RECOVER DATABASE UNTIL TIME statement, 12-28
RECOVER FROM...STANDBY DATABASE command, 12-37, 12-43
RECOVER option
STARTUP command, 12-4
recovered tablespaces
backing up, 13-16
RECOVER...FROM parameter
ALTER DATABASE comman, 12-5
recovery, 9-8
after control file damage, 12-52
after using RESETLOGS option, 3-10
applying redo logs during, 12-4
cancel-based, 4-7
procedures, 12-22
change-based, 4-7, 12-31 to 12-35
closed database, 12-3
commands, 12-11 to 12-12
Recovery Manager, 8-37
database startup, during, 12-4
datafile
guidelines for, 8-36
without backup, 12-13
distributed database, 12-2
with snapshots, 12-3
dropped table, 12-54
from media failure
examples, 12-45 to 12-54
Import utility, 11-14
interrupting, 12-9
lost control file, 12-51
media, 4-7, 12-1

complete, 4-7
control files and, 2-15
disabled, 2-16
enabled, 2-16
incomplete, 4-7
offline tablespaces in open database, 12-4
parallel processes for, 12-11
PARALLEL_MAX_SERVERS parameter, 12-11
planning, 4-1 to 4-7
point-in-time, 4-6
database, 9-10
preparing for, 12-11 to 12-15
privileges required, 12-15
procedures for complete, 12-15 to 12-21
procedures for incomplete, 12-21 to 12-35
read-only tablespaces, 12-44
recover command, 9-8
restarting, 12-9
roll forward phase, 12-4
setting number of processes to use, 12-11
SQL commands for, 12-4
strategies, 4-1 to 4-7
testing, 4-6
time-based, 4-7, 12-26 to 12-30
types
distributed database system and, 12-2
unrecoverable objects and, 12-43 to 12-44
using Import utility, 11-14
when database is open, 9-8
recovery catalog, 6-2 to 6-6, 7-5, 8-2 to 8-9
backup, 6-6
complex queries using **report**, 9-12
connecting to Recovery Manager with, 6-8
connecting to Recovery Manager without, 6-7
definition, 4-4
deleting record, 8-7
installing, 8-2
log switch record, 8-6
operating without, 7-7
querying, 8-12, 9-11
recovery of lost or damaged, 8-9
registering databases, 8-2, 8-3
resynchronizing, 6-6, 8-4
schema, 6-2
setting up, 6-5
space requirements for, 6-3
updating, 8-4
after schema changes, 8-5
Recovery Manager, 5-2, 8-19
backup
fractured block detection, 7-26
scenario, 8-27
backup types, 7-10
command file, 6-10
commands
allocate channel, 8-10, 8-28, 8-36
allocate channel for delete, 8-7, 9-8
backup, 8-21, 9-7
catalog, 8-2, 8-8
change, 8-6, 8-7
change...delete, 8-8
copy, 8-28
delete script, 8-19
execute script, 8-19
list, 8-12, 9-11
list backupset, 8-18
list copy, 8-18
list incarnation of database, 8-4
print script, 8-19
recover, 8-35, 9-8
register database, 8-3
release channel, 8-11
replace script, 8-19
replicate, 8-31
report deletable, 8-15
report need backup, 8-13
report need backup days, 8-14
report schema, 8-16
report unrecoverable, 8-14
reset database, 8-4
reset database to incarnation, 8-4
restore, 9-8
resync catalog from backup controlfile, 8-9
set newname, 8-32, 8-35
setlimit channel, 8-11
switch, 8-35
connecting with password file, 6-8
connecting with recovery catalog, 6-8
connecting without password file, 6-7
connecting without recovery catalog, 6-7

connection options, 6-7
copying datafiles with, 9-5
deleting file records, 8-7
deleting files, 8-7
deleting recovery catalog record, 8-7
fractured block detection in, 7-26
general connect, 6-7
interactive use of, 6-9
introduction, 7-2
recover command, 4-2
recovery catalog, 4-4
recovery catalog required, 8-2
recovery commands, 8-37
registering databases, 8-3
resetting database information, 8-4
running commands, 6-9 to 6-11
sample scripts, 6-11
scenarios, 9-1 to 9-12
setting time parameters, 6-11
stored scripts, 6-10, 7-9
types of backups, 8-22
user tags for backups, 7-24
recovery sets, 10-3
containing whole tables, 10-4
copying to primary database, 13-15
importing into primary database, 13-15
RECOVERY_PARALLELISM initialization parameter, 12-11
redo entries
content of, 2-2
redo log
definition, 4-3
listing log files for backup, 11-2
redo log buffers
writing of, 2-2
redo log files, 1-5
"fuzzy" data in backups and, 3-9
active (current), 2-4
archived
advantages of, 2-13
automatic, 2-17
contents of, 2-15
control files and, 2-15
errors in archiving, 2-18
log switches and, 2-5
manually, 2-18
mechanics of archiving, 2-14
archived logs, 1-7
available for use, 2-3
contents of, 2-2
distributed transaction information in, 2-3
files named in control file, 2-19
groups, 2-10
inactive, 2-4
log sequence numbers of, 2-5
log switches, 2-5
log writer process, 2-3
manual archiving, 2-18
members, 2-10
mirrored
archiver process (ARCH) and, 2-14
if all inaccessible, 2-11
if some inaccessible, 2-10
log switches and, 2-10
multiplexed, 2-9
diagrammed, 2-10
purpose of, 1-5
naming, 12-5
online, 2-2
after checkpoint failure, 2-8
recovery use of, 2-2
requirement of two, 2-3
threads of, 2-13
overview, 1-5
redo entries, 2-2
redo record
definition, 4-3
redundancy option
list command, 8-15, 8-16
register database command, 8-2, 8-3
registering
archivelog, 8-2
datafile, 8-2
release channel command, 8-11
replace script command, 8-19
replicate command, 8-31
REPORT command, 7-23
report command, 9-12
days option, 8-15
device_type_list option, 8-13

incremental option, 8-14
report_object_list option, 8-13
using for complex recovery catalog queries, 9-12
REPORT DELETABLE command, 7-23
report deletable command, 8-15
report generation, 7-23
REPORT NEED BACKUP command, 7-23
report need backup command, 8-13
report need backup days command, 8-14
report schema command, 8-16
REPORT UNRECOVERABLE command, 7-23
report unrecoverable command, 8-14
report_object_list option
report command, 8-13
reports
generating, 8-12
reset database command, 8-4
reset database to incarnation command, 8-4
RESETLOGS option, 12-3, 12-24 to 12-26
ALTER DATABASE command
backing up control file, 11-11
database backups after using, 3-5
OPEN DATABASE, 3-10
recovery of database after using, 3-10
restore command, 9-8, 9-10, ?? to 9-11, ?? to 9-11
from backup operand, 8-33
from copy operand, 8-33
restore constraints, 7-26
restore destination
control file, 8-31
datafile, 8-31
restore specification, 8-34
list, 8-34
restoring, 9-8
backup file selection, 8-30
datafile
guidelines, 8-32
when database is open, 9-8
whole database backups, 12-9
restoring datafiles, 8-30
resuming recovery after interruption, 12-9
resync catalog from backup controlfile command, 8-9
resynchronizing recovery catalog, 6-6
rollback segment, 1-6
definition, 4-3
rollback segments
and TSPITR, 10-4
rolling forward, 1-5

S

scenarios
Recovery Manager, 9-1 to 9-12
schema changes
updating recovery catalog, 8-5
scripts
catrman.sql, 8-3
scripts, stored
creating, 8-19
deleting, 8-19
printing, 8-19
Send Us Your Comments
boilerplate, xiii
sequence change number(SCN)
format, 12-32
sequential media
backups to, 5-5
SET AUTORECOVERY ON statement, 12-6
SET command
LOGSOURCE parameter, 12-5
set newname command, 8-30, 8-32, 8-35
Set Tablespace Offline dialog, 11-9
Set Tablespace Online dialog, 11-9
set until limit
setlimit channel command, 8-30, 8-36
setlimit channel command, 8-11
set until limit, 8-30, 8-36
setmaxcorrupt clause
backup command, 9-7
setsize parameter, 8-24
shutdown
checkpoints and, 2-6
SHUTDOWN command
ABORT option, 12-10, 12-22, 12-26, 12-52
Shutdown Database dialog box
Abort mode, 12-22, 12-26
Enterprise Manager, 12-52
Shutdown Abort mode, 12-10

skip inaccessible option
backup command, 9-7
skip offline option
backup command, 9-2, 9-7
skip read only option
backup command, 9-2
skipping tablespaces during backup, 9-2
snapshot control file, 7-9
snapshot tables
and TSPITR, 10-5, 13-4
snapshots
distributed database recovery and, 12-3
media recovery and, 12-3
SQL commands
applying log files, 12-6
for recovering tablepace, 12-6
recovery using, 4-2, 12-4
standby database, 12-35 to 12-44
activating, 12-38
control file for, 12-36
creating, 12-35, 12-36
effect of altering primary database, 12-39
maintaining, 12-36
mounting, 12-38
starting a database
recovery during, 12-4
STARTUP command
MOUNT option, 12-23, 12-27
RECOVER option, 12-4
Startup Database dialog box, 12-27
statement failure, 1-3
stored scripts, 7-9, 8-19
creating, 8-19
deleting, 8-19
printing, 8-19
Recovery Manager, 6-10
switch command, 8-35
system change number (SCN)
definition, 4-3
use in distributed recovery, 12-3
when determined, 2-2
system failure, 1-3
SYSTEM tablespace
media failure in ARCHIVELOG mode, 12-46
recovery of, 12-17
system time, changing
effect on recovery, 12-21

T

table
dropped
recovery, 12-54
tablespace option
list command, 8-17
TABLESPACE parameter
RECOVER command, 12-4
tablespace point-in-time recovery
and recovery sets containing whole tables, 10-4
introduction, 10-2, 13-2
limitations, 10-4, 13-4
performing, 10-9, 13-1 to ??, 13-7, ?? to 13-30
planning for, 10-3, 10-7, 13-3
requirements, 13-6
restrictions, 10-6, 13-5
tuning considerations, 10-10, 13-28
backup control files, 13-29
recovery set location, 13-28
tablespaces
backing up, 9-2, 9-3
backing up after TSPITR, 10-10
backing up offline, 11-8
backing up online, 11-5
backing up several online, 11-8
backups and checkpoints, 2-6
frequency of backups, 3-3
listing backup sets including datafiles from, 8-17
read-only
backing up, 9-2
effect on recovery, 12-44
recovering offline in open database, 12-4
restoring
when database is open, 9-8
skipping during backup, 9-2
tag operand
backup command, 8-23
list command, 8-18
threads
online redo log, 2-13

time format
RECOVER DATABASE UNTIL TIME statement, 12-28
time parameters, setting for Recovery Manager use, 6-11
time-based recovery, 4-7, 12-26 to 12-30
archived redo log, 12-51
coordinated in distributed database, 12-2
trace file
ARCH, 2-18
control file backups to, 11-10
log writer process and, 2-11
transactions
committing
writing redo log buffers and, 2-2
rollback, 1-6
uncommitted, 1-6
TS_PITR_CHECK view, 10-4, 13-4
TSPITR. See tablespace point-in-time recovery.
type disk operand
allocate channel command, 8-10

U

unavailable operand
change command, 8-7
uncatalog operand
change command, 8-7
unrecoverable objects
and RECOVER operation, 12-43
recovery and, 12-43 to 12-44
user errors
database failure and, 1-3
recovery from, 12-53
user tags, 7-24
user-created backup file
cataloging, 8-8
USING BACKUP CONTROLFILE parameter, 12-23, 12-32
utilities
OS, using to make copies, 9-7
Recovery Manager, 7-2

V

V$BACKUP, 11-7
V$BACKUP_CORRUPTION, 8-21
V$BACKUP_DEVICE, 9-2
V$DATAFILE, 12-4, 12-28, 12-32
listing files for backup, 11-2
V$DATAFILE_HEADER, 9-9
V$LOGFILE, 12-47
listing files for backup, 11-2
V$LONGOPS, 8-21
V$RECOVER_FILE, 12-12
views
data dictionary, 11-5
DBA_DATA_FILES, 11-5
V$BACKUP, 11-7
V$BACKUP_CORRUPTION, 8-21
V$DATAFILE, 11-2, 12-4, 12-28, 12-32
V$LOGFILE, 11-2, 12-47
V$LONGOPS, 8-21
V$RECOVERFILE, 12-12

W

warning
archiving mode for first backup, 3-8
consistency and Export backups, 11-13
whole database backups, 11-3 to 11-4
consistent, 2-22
definition, 2-21
inconsistent, 2-23
restoring from, 12-9